National

パーソナルコンピュータ

品番

MSX CF-2000

取扱説明書

National

パーソナルコンピュータ

MSX CF-2000

取扱説明書

はじめに

このたびはナショナルパーソナルコンピュータ CF-2000をお買い上げいただきまして，まことにありがとうございました．

CF-2000は，MSX システムとして幣社が最新技術を用いて開発したもので，手軽な操作性と機器の拡張性等，本格的なパーソナルコンピュータとして充分ご満足いただけるものと確信致しております．

本書には，ご使用の場合の取扱い事項をくわしく掲載致しておりますので，ご活用ください．なお，「BASIC」プログラムのご使用については別冊の「BASIC 説明書」をも併せてご参照ください．

本機を正しくご使用いただくために，本書を大事に保管し，末長くご愛用いただきますようお願い申し上げます．

MSX システムは，メーカー，機種を問わないソフトウェアの自在な互換性の夢を実現したパーソナルコンピュータです．

本書の使い方

本書は，入門編，解説編，資料編の3つに分かれています．

入門編では，本機をはじめて動かすときのセットアップの仕方や，BASICやコンピュータのことをまったく知らない人のために初歩的な操作について解説してあります．

解説編では，MSX BASICを使っていて疑問や問題点にぶつかったときの対処の仕方や，コンピュータ用語，コンピュータの仕組みなどについてコラム風にまとめてあります．読みものとして読んでも役に立つことがあるでしょう．

資料編は，MSXシステムのハードウェア，ソフトウェアに関する表や図の集まりです．MSXシステムをより有効に使うために必要となるものです．

はじめて本書を見るとき，とても難しいと思われるかもしれませんが恐れることはありません．BASICという言語は，初心者は初心者なりに使えるものですから一度にすべてを覚える必要はないのです．キーボードをどのように操作しても，どんなプログラムを実行してもMSXが壊れてしまうことはありません（キーボードを力いっぱいたたくなどは別の話）．とにかくいろいろな実験をして，早くBASICに慣れてしまうことが大切です．MSXを実際に使いながら，本書を参考にするようにしてください．

目 次

入門編

解説編

資料編

INDEX

入門編

1

取扱上のご注意

下記の注意事項に従いお使いください. 使い方を誤ると故障の原因になりますのでご注意ください.

■カートリッジの抜差は電源OFFの状態で行なってください. またスロットのふたは手で開けないでください. 通電中にスロットのふたを開けますと実行中のプログラムが壊れます.

■高温, 低温, 直射日光およびホコリの多い場所での使用はさけてください. (変色する場合があります). また急激な温度変化はさけてください.

■コーヒー, ジュース, 紅茶などの飲物をこぼさないでください.

■分解しないでください. 異常が感じられたときは, 販売店にご相談ください.

■お手入れはシンナー, ベンジンなどの揮発性液体の使用はさけ, 乾いた布でふいてください.

■落としたり, ぶつけたり, 強いショックを与えないでください.

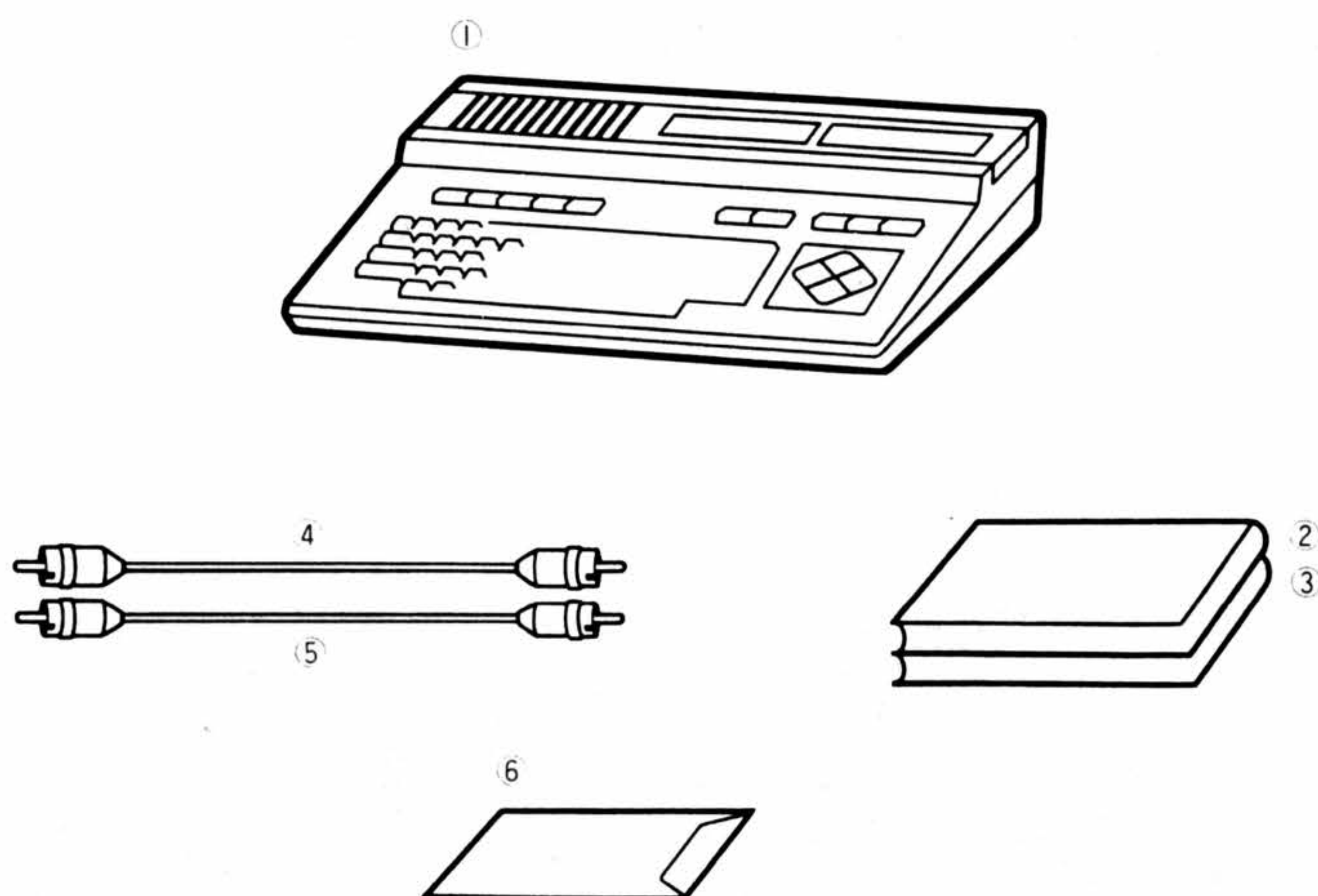
■ラジオなどの受信機の近くで使用しますと受信機に雑音が入ることがありますので, はなしてご使用ください.

■長時間使用されないときは, 本体の電源スイッチを切り電源プラグを抜いておいてください.

■指定外の機器を本機に接続することは, 絶対におやめください.

A. パッケージの説明

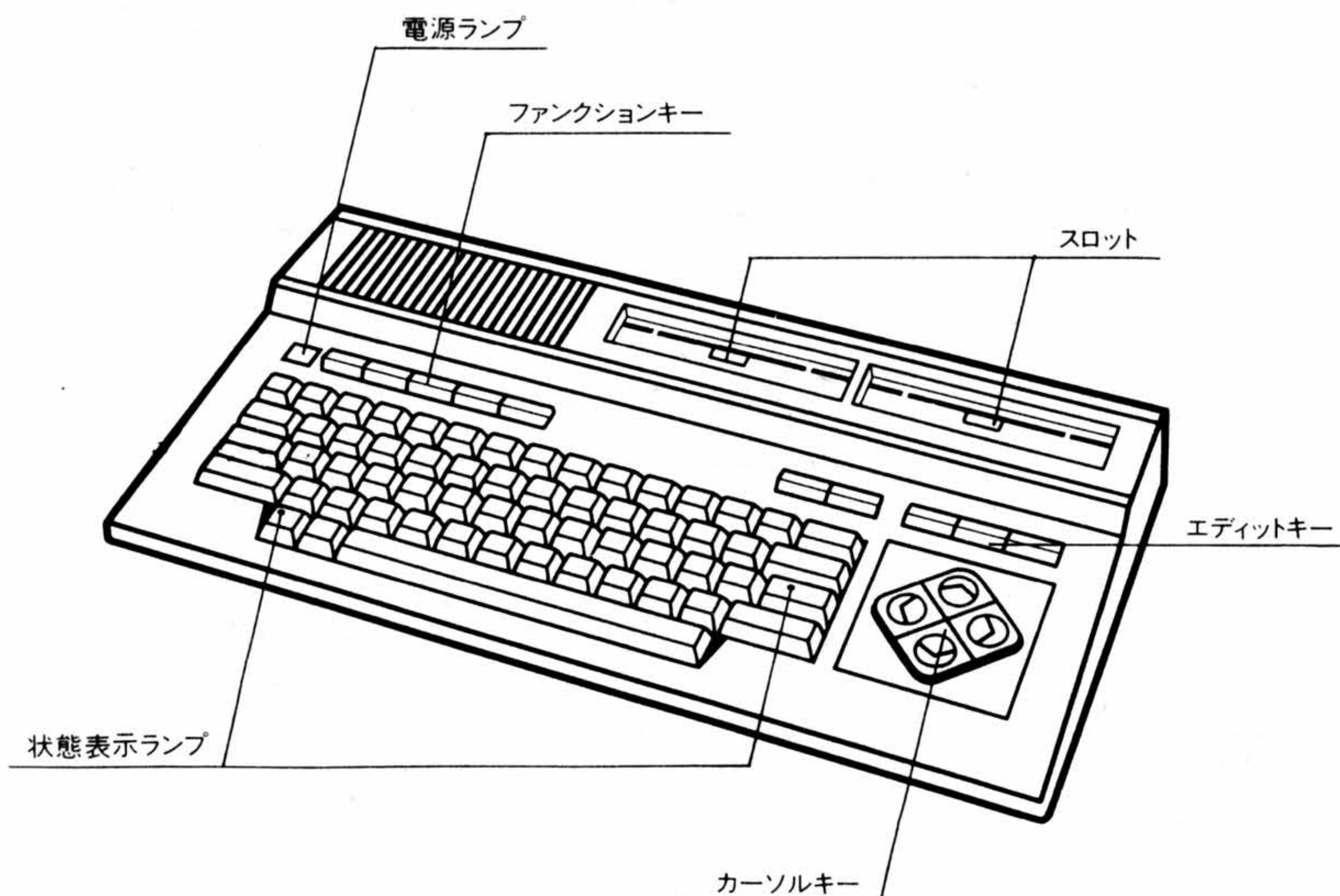
開梱を行い，内容物を取り出してください．内容物には次のものが含まれています．万一欠品がございましたらお買い求めの販売店まで御連絡願います．

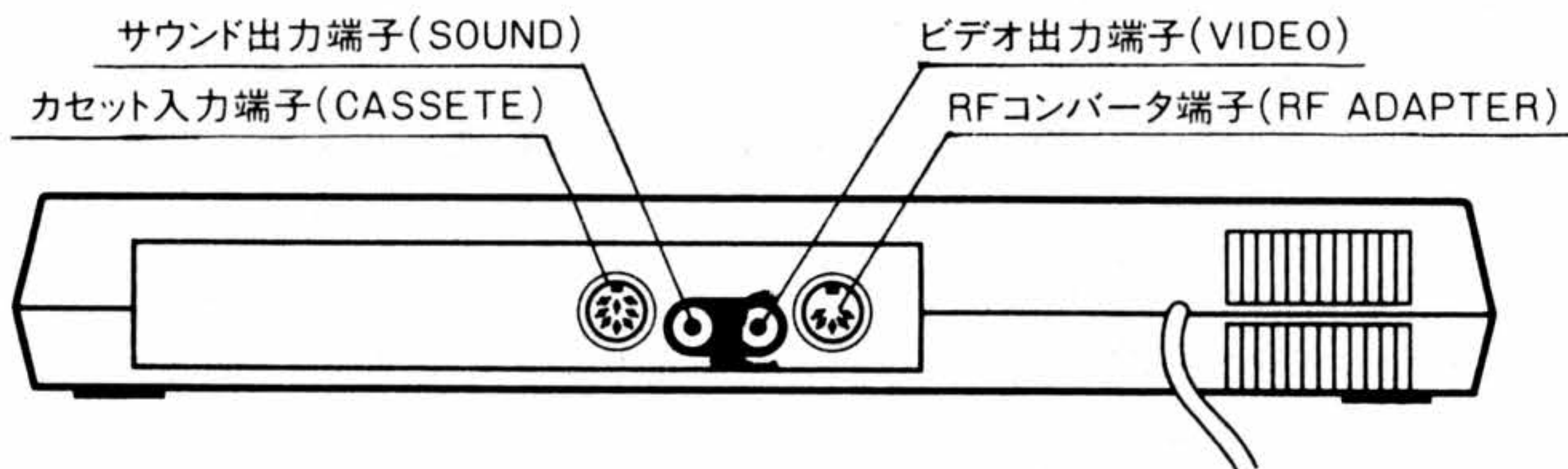
① CF-2000 本体
② CF-2000 取扱説明書（本書）
③ CF-2000BASIC 説明書
④ ビデオケーブル
⑤ サウンドケーブル
⑥ 保証書

保証書は，必ず「販売店名・購入日」等の記入を確かめて販売店から受け取っていただき，内容をよくお読みの後大切に保管してください．

B. 各部の名称と機能

CF-2000の各部の名称と機能を説明します.

(1) 正面

スロット	カートリッジを挿入します.
電源ランプ	電源がONの状態のとき点灯します.
エディットキー	画面に表示されている文字の編集に使います.
カーソルキー	カーソルを上下左右に移動させるのに使います.
状態表示ランプ	それぞれ, かなモード, CAPSモードであるとき点灯します.
ファンクションキー	一度押すと, 定義された文字列がキー入力されます.
キーボード	命令などをキー入力するのに使います.

B. 各部の名称と機能

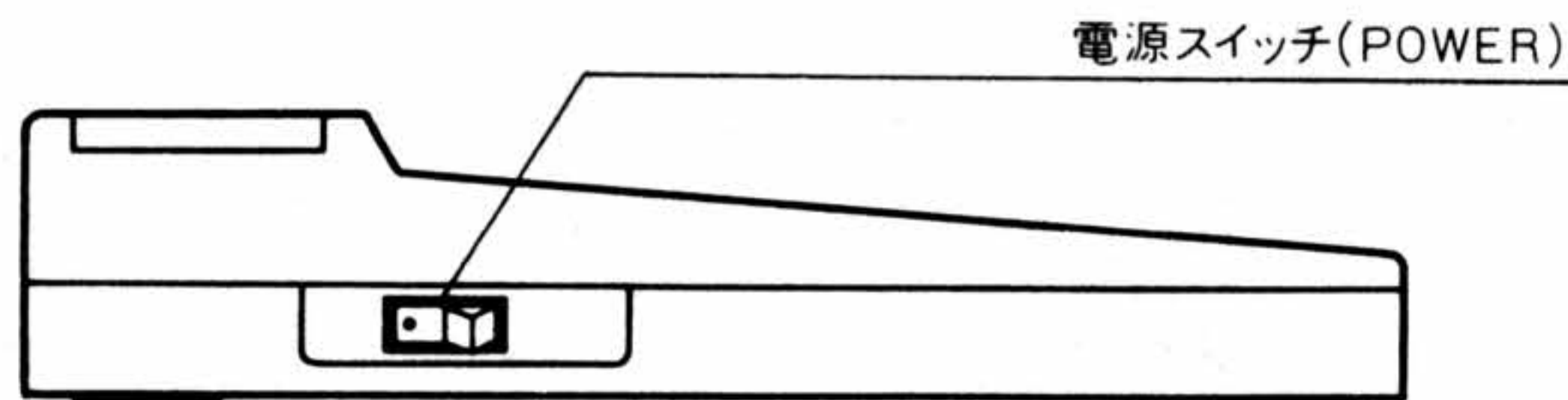
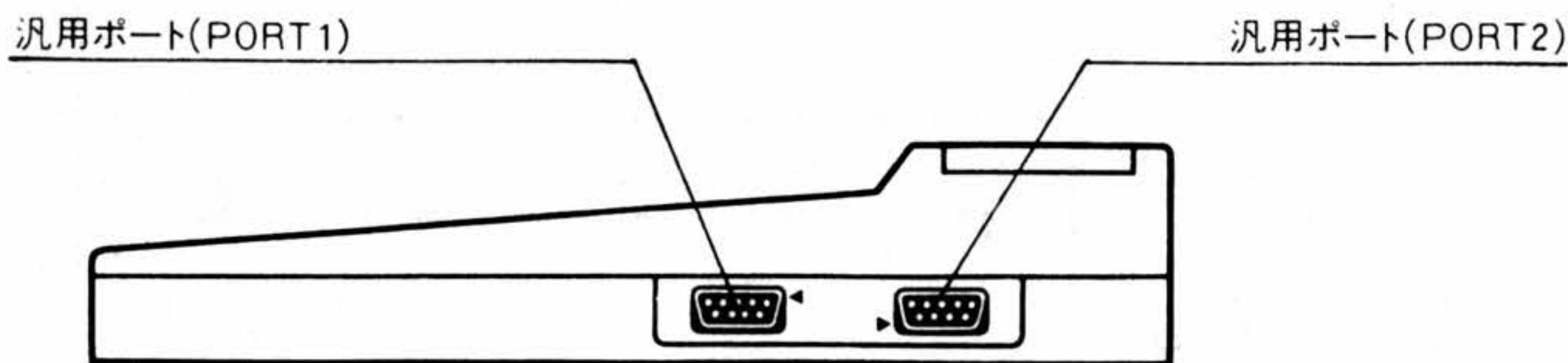
(2) 背面

カセット入力端子　カセットレコーダを本体に接続するための端子です.

ビデオ出力端子　ディスプレイ装置に文字や絵を出力するための端子です.

サウンド出力端子　ディスプレイ装置に音を出力するための端子です.

RF コンバータ端子　家庭用 TV 受像機を本体に接続するための端子です.

電源コード　AC100 V の電源にプラグを差し込みます.

(3) 側面

電源スイッチ　電源コードがプラグに差し込まれているとき，電源スイッチをONにすると電源が入ります.

汎用ポート端子　ジョイスティックまたはタブレットを2台まで接続することができます.

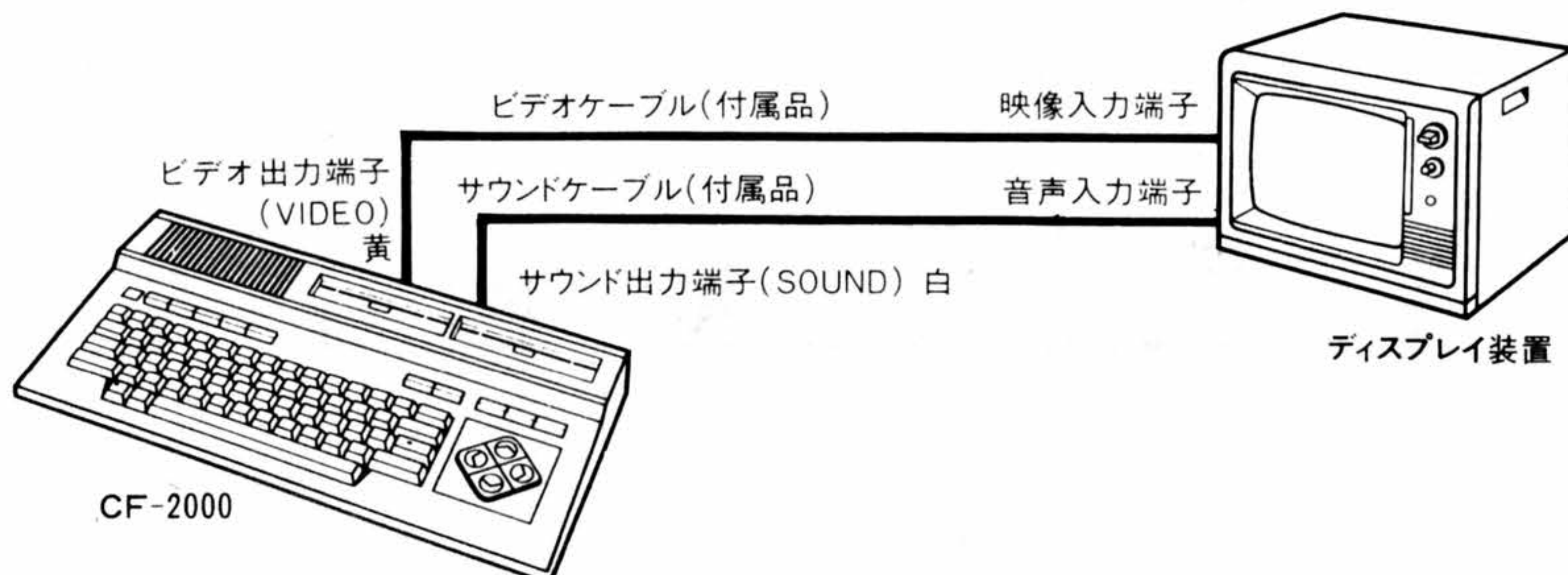
C. セットアップ

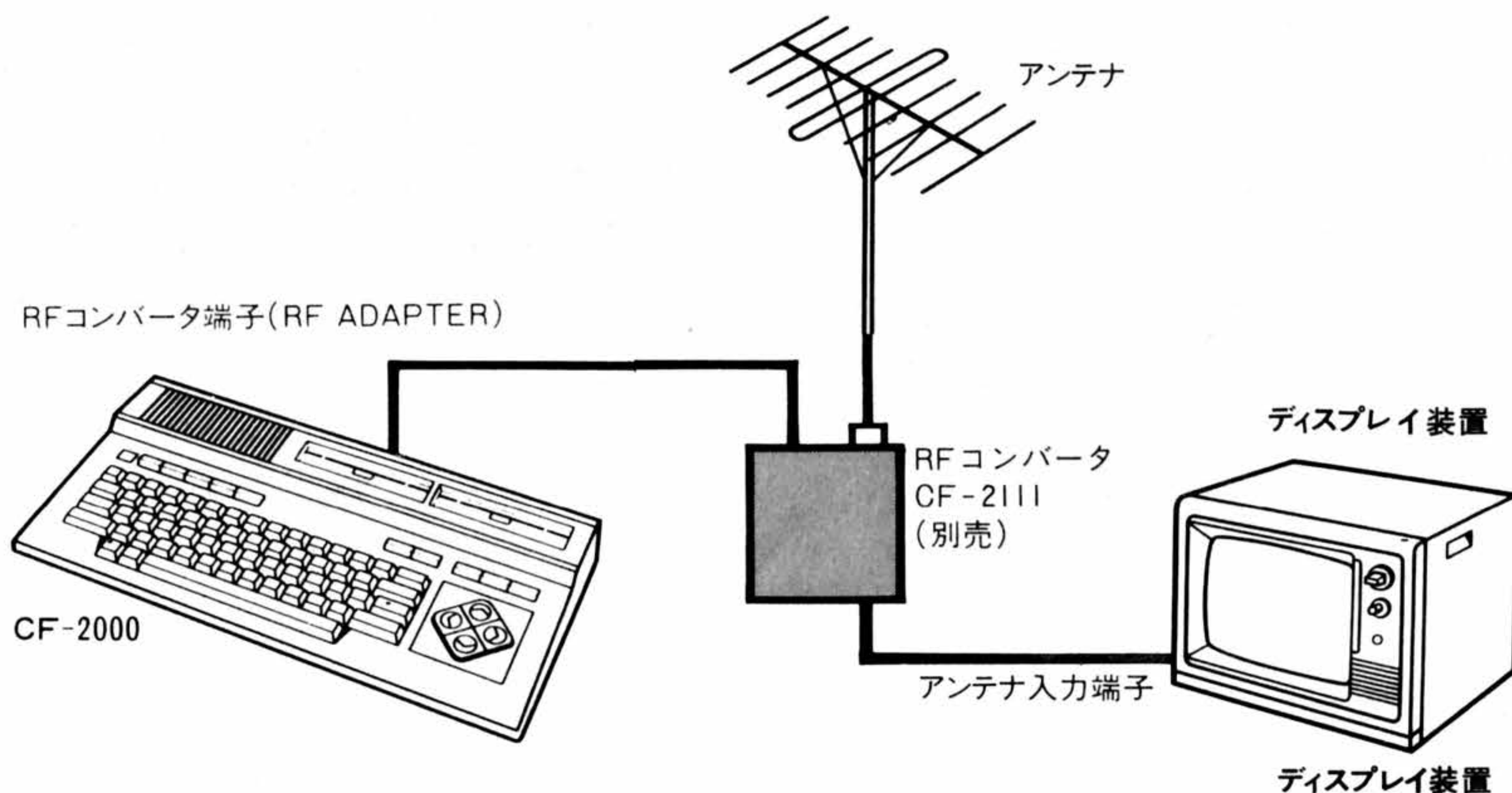
(1) マシンの設置

本機を設置する際には次の注意事項を守ってください.

- 直射日光の当る場所や発熱する器具の近くで使用することは避けてください.
- 極端に湿気の多い場所や，ほこりの多い場所で，使用および保管することは避けてください.
- 衝撃を加えたり，衝撃，振動の加わる場所での保管や使用は避けてください.
- 本機の上に重い物を置くことは避けてください.
- ラジオやテレビなどの近くで使用しますと，ラジオやテレビに雑音が入ることがあります．また，強い磁界を発生する装置などが近くにありますと，逆に本機に雑音が入ることがあります．なるべく離してご使用ください.

(2) ケーブルの接続

本機をご使用になるには，ディスプレイ装置を本機に接続する必要があります．ビデオ入力端子付の場合は,付属のビデオケーブルとサウンドケーブルを使います．ビデオケーブル は一方の端子を本体のビデオ出力端子に挿入し，もう一方の端子をディスプレイ装置の映像入力端子に挿入します．サウンドケーブルも，一方の端子を本体のサウンド出力端子に挿入し，もう一方の端子をディスプレイ装置の音声入力端子に挿入します．ケーブルの端子は左右対称です.

C. セットアップ

なお，ディスプレイ装置として，ビデオ入力端子のない家庭用TV受像機をご使用になる場合は，オプションのRFコンバータ(CF-2111)をお買い求めになり，付属の取扱説明書をご参照の上TVのアンテナ端子と接続してください.

画面の文字が見にくいときは，コントラストつまみを調整してみてください.

外部記憶装置として，カセットレコーダをご使用になる場合は，(20)ページの「オーディオカセットレコーダ」を参照してください.

(3) 電源投入

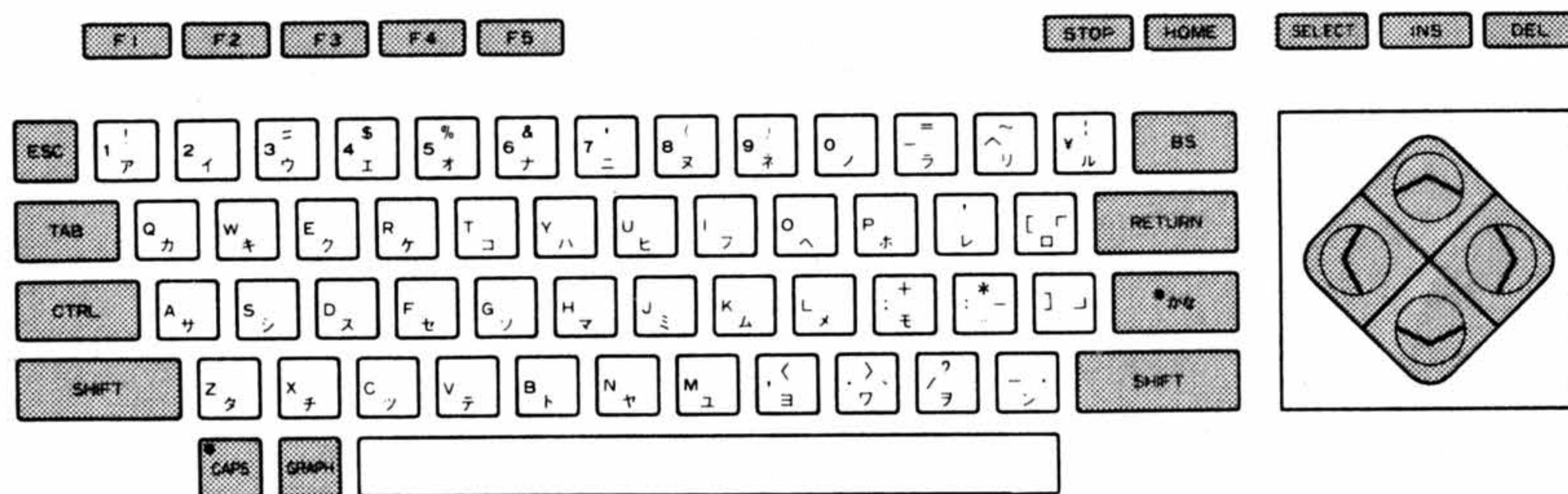
まず，ディスプレイ装置の電源を入れ，画面を表示できる状態にしてください．それから本体の電源プラグをAC 100Vの電源に差し込み，電源スイッチをオンにしてください.

本体の電源ランプが点灯し，ディスプレイ画面には次のように表示されます.

```
MSX BASIC version 1.0
Copyright 1983 by Microsoft.
12431 Bytes free
Ok
```

この時点で，BASICモードとなり，BASICの命令をキーボードから入力することが可能となります．テレビの画面の一番左端の文字が見にくいときは，WIDTH命令を使ってみてください(解説編54ページ参照).

D. キーボード

本機のキーボードは次のような配置になっています.

図の で示したキーは特殊キーと呼び, その他のキー (文字キー) と区別しています. まず, 特殊キーについて説明しましょう.

● 特殊キー

SHIFT 　英大文字 (ただし CAPS がロックされているときは英小文字), かな小文字または文字キーの上段の文字をタイプするのに使います. このキーはキーボードの左右両側についていますが, どちらを使っても構いません.

CAPS 　英大文字をタイプするのに使います. 一度押すとロックされた状態となり, キートップの赤いランプがつきます. もう一度押すとロックは解除されます. ロックされた状態で英字キーを押すと大文字がタイプされ, 解除された状態では小文字がタイプされます.

かな 　かな文字をタイプするのに使います. 一度, 押すとロックされた状態となり, キートップの赤いランプがつきます. もう一度押すとロックは解除されます. ロックされた状態で, 文字キーを押すと, ひらがながタイプされます. このとき CAPS がロックされているとカタカナがタイプされます.

GRAPH 　グラフィック記号をタイプするのに使います. このキーを押しながら文字キーを押すとグラフィック記号がタイプされます.

CTRL 　他のキーと組み合わせて, 特殊な機能を実行します.

D. キーボード

それぞれカーソルを上下左右に動かすのに使います.
本書では [↑][↓][←][→] として説明しています.

[INS]
このキーを押すと挿入モードになり，カーソルの形が変わります．挿入モードで文字キーを押すと，カーソルの直前に文字が挿入されていきます．再びこのキーを押すかカーソル移動キーや [RETURN] キーを押すと，挿入モードが解除されて，カーソルも元の形に戻ります.

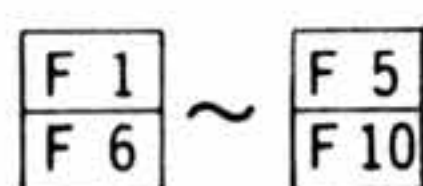
[DEL]
カーソル上の文字が削除されます．また，カーソルの右側の文字列はすべて左に一桁移動します.

[BS]
カーソルの直前の文字が削除されます．また，カーソルの位置から右側の文字はすべて左に一桁移動します.

[TAB]
このキーを押すとカーソルは次のTAB位置まで進み，その間の文字列は空白となります．TAB位置は1，9，17，25と8桁ごとにセットされています.

[HOME/CLS]
カーソルが画面左上端に移動します.
[SHIFT] キーを押しながら [HOME/CLS] キーを押すと，カーソルが画面左上端に移動すると共に，画面がクリアされます.

[STOP]
このキーを押すとプログラムはストップし，一時停止状態となります．再びこのキーを押すと処理は継続します．この一時停止状態のとき，キー入力することはできません．キー入力したい場合は，[CTRL] キーを押しながら [STOP] キーを押してください．このとき，処理を継続するには [C]，[O]，[N]，[T]，[RETURN] とキー入力します.

[RETURN]
キー入力した文字列をコンピュータに送るのに使います.

[F1/F6]～[F5/F10]
ファンクションキーと呼ばれ，各キーに対応した文字列がタイプされます.
[SHIFT] キーを押しながら押すと [F6]～[F10] として働きます.

[SELECT]
プログラム中で使用します．コントロールコード24を入力できます.

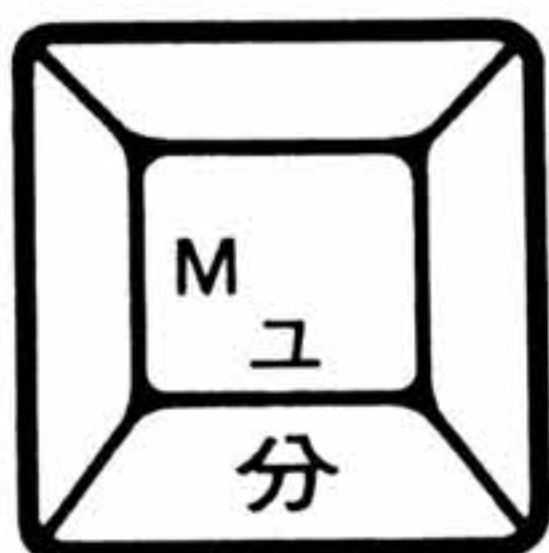
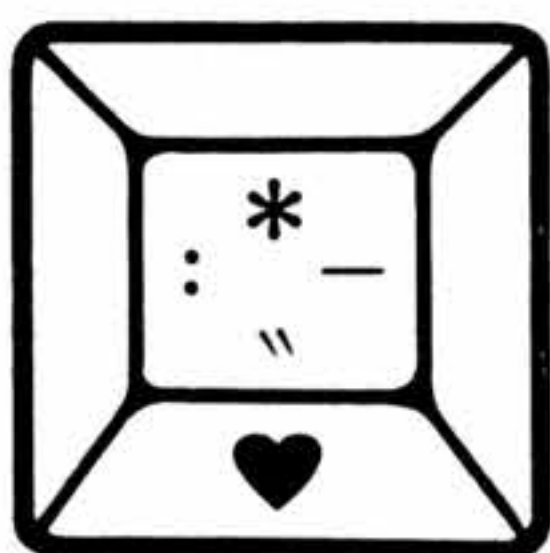
[ESC]
プログラム中で使用します．コントロールコード27を入力できます.

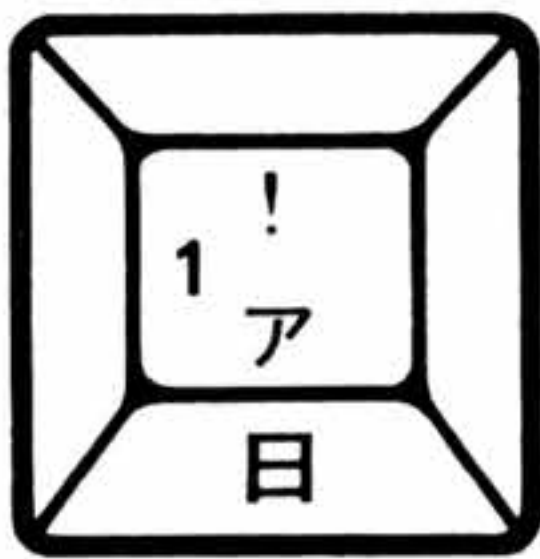
D. キーボード

● 文字キー

1つの文字キーで数種類の文字をタイプすることができます．どの文字をタイプするかは特殊キーと組合せで選択することができます．

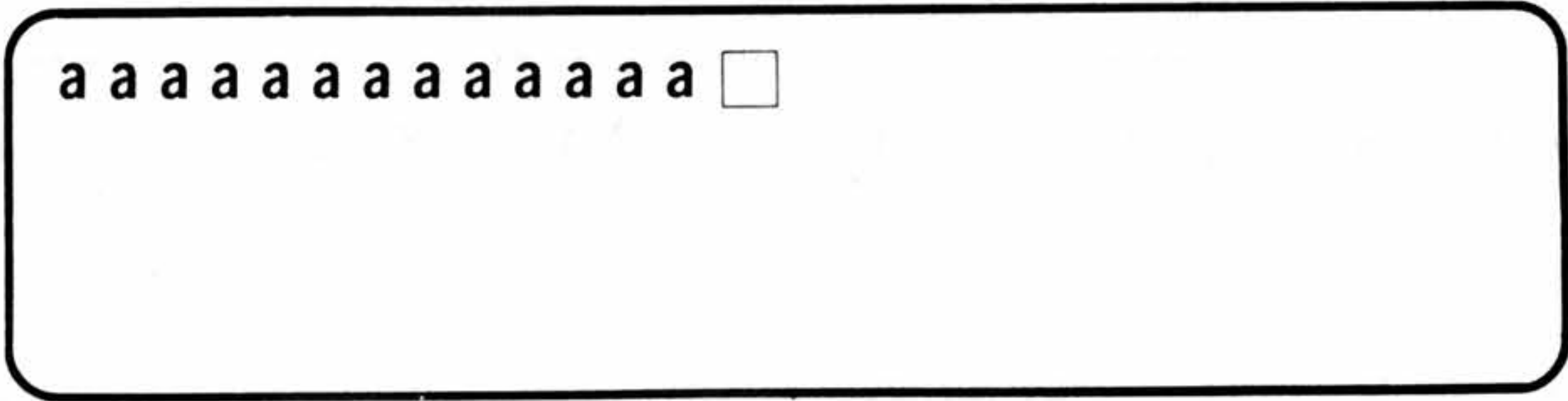
ここで〝+〟と〝，〟は次のことを意味します．

A + B	A キーを押しながら B キーを押す
A，	A キーを1度押して，A キーのモードにしておく

押すキー	タイプされる文字
M ユ	m（英小文字）
SHIFT + M ユ	M（英大文字）
CAPS，M ユ	M（英大文字）
かな，M ユ	ゆ（ひらがな）
かな，SHIFT + M ユ	ゅ（ひらがな小文字）
かな，CAPS，M ユ	ユ（カタカナ）
かな，CAPS，SHIFT + M ユ	ュ（カタカナ小文字）
GRAPH + M ユ	分（グラフィック）

押すキー	タイプされる文字
: * _ 〃	:
SHIFT + : * _ 〃	*
かな，: * _ 〃	〃
かな，SHIFT + : * _ 〃	―
GRAPH + : * _ 〃	♡

D. キーボード

押すキー	タイプされる文字
1!ア	1
SHIFT + 1!ア	!
かな, 1!ア	あ
かな, SHIFT + 1!ア	ぁ
かな, CAPS, 1!ア	ア
かな, CAPS, SHIFT + 1!ア	ァ
GRAPH + 1!ア	日

● オートリピート機能

同じキーを押し続けると文字が連続してタイプされます．たとえば A キーを押し続けるとａが連続してタイプされます．

```
aaaaaaaaaaaaaa□
```

このような機能をオートリピート機能と呼んでいます．この機能は文字キーばかりでなく特殊キー（カーソル移動キー，DEL キーなど）にも適用されます．

E. 文字の編集

● 文字の入力

キーボードの文字キーを押すと，対応する文字が画面のカーソル位置に表示されます．

[A] から [Z] までの文字をタイプしてみてください．

```
abcdefghijklmnopqrstuvwxyz□
```

● 文字の置換

ここで，def をそれぞれ大文字に変えてみましょう．そのためには次のようにしてください．

- カーソルを [←] キーを使って d の位置へ戻す
- [SHIFT] キーを押しながら [D] [E] [F] とタイプする

画面は次のように変わります．

```
abcDEF[g]hijklmnopqrstuvwxyz
```

このように，文字を置き換えたい場合は，カーソルを該当文字位置に動かし，新たな文字をタイプすればよいのです．

E. 文字の編集

● 文字の削除

次に，ghi の 3 文字を削除してみましょう．まず，DEL キーを押してください．カーソル上の文字 g が削除されます．

```
abcDEF[h]ijklmnopqrstuvwxyz
```

次に，DEL キーを続けて 2 回押すと，今度は h と i が削除されます．

```
abcDEF[j]klmnopqrstuvwxyz
```

このようにカーソルの後方（右方向）の文字を削除するには DEL キーを続けて押せばよいのです．

今度は DEF の 3 文字を削除してみましょう．BS キーを押してください．カーソルの直前の文字 F が削除されます．

```
abcDE[j]klmnopqrstuvwxyz
```

さらに続けて 2 回 BS キーを押すと，E，D の順に文字が削除されます．

```
abc[j]klmnopqrstuvwxyz
```

このようにカーソルの前方（左方向）の文字を削除するには BS キーを続けて押せばよいのです．

E. 文字の編集

● 文字の挿入

今度はcとjの間にdefghiの6文字を挿入してみましょう．まず INS キーを押してください．するとカーソルの形が■から＿に変わり，挿入モードになります．ここで D とタイプすると，カーソルの直前にdが挿入されます．

```
abcdjklmnopqrstuvwxyz
```

続けてefghiとタイプしてください．

```
abcdefghijklmnopqrstuvwxyz
```

“efghi”がdとjの間に挿入されます．最後，再び INS キーを押すと，挿入モードが解除されて，カーソルも元の形に戻ります．

F. BASIC

人間どうしで会話をするのに日本語，英語などの特有な言語が必要なように，人間とコンピュータが会話するにも特別な言語が必要です．この人間とコンピュータが会話するための言語がBASICです．

BASIC言語の詳細な説明はBASIC説明書を参照してください．ここでは，BASICの基本的な操作方法について説明します．

● BASIC命令の実行

BASICの命令は，キーボードからキー入力したあと，最後に RETURN を押すと，コンピュータに送られて実行されます．

まず，指定した文字列や数字をディスプレイ画面に表示するPRINT命令を使ってみましょう．最初に CLS キーを押して，画面をクリアしてから，P R I N T スペースキー 1 0 と順にキーを押し，最後に RETURN キーを押してください．次のように画面に表示されます．

```
PRINT 10 ………キー入力した命令
  10 …………………………命令の実行結果
Ok ……………………………命令の実行が完了したことを示す
□ ……………………………カーソル
```

PRINT 10は「画面に10を表示せよ」という命令です．この命令の実行結果である10が命令のすぐ下に表示されています．**命令の最後で RETURN キーを押すことを忘れないでください． RETURN キーを押すことによりはじめて命令がコンピュータに送られますから，押し忘れるとコンピュータはウンともスンとも言いません．**以降の説明では〝↵〞で表示します．またBASICの命令は大文字も小文字も同じですからPRINTのかわりにprintとしてもかまいません．

● 文字の表示

PRINT命令で文字列を表示させたい場合は，文字列の前後をダブルクォーテーションマーク（″）で囲んでください．たとえば「basic」と表示させたい場合は次のようにします．

```
? "basic"
basic
Ok
□
```

ここではPRINT命令の省略形である？を使っています．

F. BASIC

● 式の値の表示

PRINT 命令で計算式を指定することもできます．これにより，コンピュータを電卓代わりに使うことができます．その例をいくつか示しましょう．

```
?  10+5 ..................... 10+5
  15
Ok
?  15*3+8/4 ................. 15×3+8÷4
  47
Ok
?  100-5^2 .................. 100-5²
  75
Ok
?  (10+2)*SQR(100) ..... (10+2)×√100
  120
```

● 命令の再実行

一度 RETURN キーを押して，実行した命令を再実行させるには原則として，命令を再び入力しなければなりません．しかし，命令が画面に残っている間は再入力する必要はありません．そのときは，カーソルを再実行したい命令のある行に移してから，RETURN キーを押せばよいのです．命令を少し変えて実行したい場合は，編集キーを使って命令を変えてから RETURN キーを押してください．たとえば

```
PRINT  10+5*2
  20
Ok
□
```

と画面に表示されているとき，10＋6×2 を計算させたければ，まずカーソルを 5 の位置に移してください．

F. BASIC

```
PRINT  10+[5]*2
  20
Ok
```

ここで 6 RETURN とタイプすれば画面は次のように変わります.

```
PRINT  10+6*2……修正された命令
  22……………………新たな結果
Ok
□
```

● BASIC プログラム

RETURN キーを押して，すぐ BASIC を実行してしまうのでは簡単な表示や計算などしかできません．もっと複雑なことをやらせるためには，いくつかの BASIC 命令を連続して実行させます．

そのためには BASIC の命令の前に 1 から65529までの番号を付けて入力します．たとえば

```
10 PRINT "B" ↵      (↵ は RETURN キーを押すこと)
```

と入力すると RETURN キーを押しても PRINT "B" は実行されずにメモリに格納されます．続けて

```
20 PRINT "A" ↵
30 PRINT "S" ↵
40 PRINT "I" ↵
50 PRINT "C" ↵
```

と入力すると，これらの命令もすべてメモリに格納されます．なお，命令の前の数字は行番号と呼ばれ，命令は行番号の若い順に実行されます．これからは行番号のついた一つの行を，その番号に従って「20行」，「30行」のように呼ぶことにします．

F. BASIC

メモリには次のように格納されているでしょう．

```
10 PRINT"B"↵
20 PRINT"A"↵
30 PRINT"S"↵
40 PRINT"I"↵
50 PRINT"C"↵
```

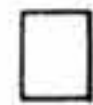
このような命令の集まりはプログラムと呼ばれています．

● プログラムを見る

現在メモリに格納されているプログラムを見るにはLISTコマンドを使います．ためしに

```
LIST ↵
```

とタイプしてください．画面には次のように表示されるでしょう．

```
10 PRINT "B"
20 PRINT "A"
30 PRINT "S"
40 PRINT "I"
50 PRINT "C"
Ok
□
```

● プログラムを実行する

メモリの中にあるプログラムを実行するにはRUNコマンドを使います．

```
RUN ↵
```

とタイプして，先ほどのプログラムを実行してみましょう．

F. BASIC

● プログラムの修正

プログラムを作っても，最初から完全なものを作るのは難しく，普通はプログラムを何度も修正しなければなりません．ここでは，プログラムの修正法を学びましょう．

● 行の挿入

プログラムの修正には，行番号が大きな役割りを持ちます．行を挿入する場合には，挿入する命令に前後の行の間の行番号をつければよいのです．たとえば20行と30行の間に次の命令を挿入するとします．

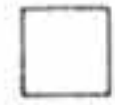
```
PRINT "I"
```

この命令には20と30の間の行番号（たとえば25）を付けてキー入力すればOKです．

```
25 PRINT "I" ↵
```

LISTを取って，確かめてください．

F. BASIC

```
LIST
10 PRINT "B"
20 PRINT "A"
25 PRINT "I"
30 PRINT "S"
40 PRINT "I"
50 PRINT "C"
Ok
□
```

● 行の削除

特定の1つの行を削除するには，その行番号のみを入力してください．たとえば，25行を削除するには

25 ↲

とタイプすればOKです．

特定の範囲の行を削除するにはDELETEコマンドを使います．たとえば20行から40行を削除するには

DELETE 20-40 ↲

とタイプします．別の例をいくつか挙げましょう．

DELETE 20	20行を削除
DELETE −40	先頭行から40行までを削除

またプログラムすべてを削除したい場合は

NEW ↲

とタイプしてください．

F. BASIC

● 行の置換え

ある行を別の行で置き換えたい場合は，同じ行番号を指定します．たとえば

```
10 PRINT "b" ↵
```

と入力すると，前の10行の内容は新しい入力内容に置き換わります．LISTを取ると次のように表示されるでしょう．

```
LIST
10 PRINT "b"
20 PRINT "A"
30 PRINT "S"
40 PRINT "I"
50 PRINT "C"
```

行の修正には，別の方法もあります．たとえば，10行を元に戻したい場合は，まず

```
LIST 10 ↵
```

として，10行を画面に表示します（すでに画面に表示されている場合はLISTをとる必要はない）．

```
10 PRINT "b"
```

ここで，カーソルをbの位置に移動させて，Bとタイプします．

```
10 PRINT "B"
```

最後に RETURN キーを押せば，10行は修正されます．これは，RETURN キーを押すと，そのつどカーソルのある行がコンピュータに入力されるという性質を利用したものです．この方法ですと，必要な個所だけを直せばよいので，行をすべて入力し直すのに比べて，簡単です．

G. オーディオカセットレコーダ

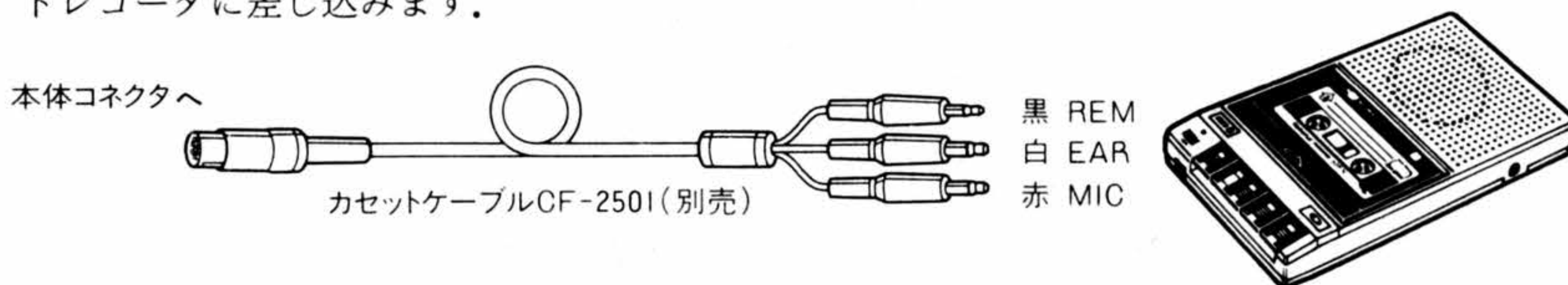
メモリ内のプログラムは本体の電源を切ると消えてしまいます．そこで，プログラムを保存したい場合は，カセットテープなどにセーブしておかなければなりません．再び，プログラムを実行させたい場合はカセットテープからメモリにプログラムをロードする必要があります．ここでは，カセットテープにプログラムをセーブしたり，ロードする方法を学びます．(オーディオカセットレコーダは以降カセットレコーダと略します．)

1　カセットレコーダの接続

まず，本体にカセットレコーダを接続しなければなりません．接続するカセットレコーダとしては次の条件を満たしているものが便利です．

- リモート端子を備えている
- テープカウンタを備えている
- モノラルの録音・再生ができる
- リモート端子を接続したまま早送り，巻戻しができる

カセットレコーダを接続する際には，本体の電源をOFFにしておきましょう．次に付属の専用ケーブルの端子を本体のカセットレコード用コネクタに差し込み，プラグをカセットレコーダに差し込みます．

オーディオカセットレコーダ

リモート端子のないカセットレコーダを使用する場合は，黒いプラグは差し込みません．
赤いプラグは録音用ですが，「唇の色だからマイク」と覚えておくと便利です．
お手持のカセットレコーダで，ロード／セーブがうまくいかないときは，RQ-8300（松下電器産業㈱製）を推奨します．

2　プログラムのセーブ

プログラムをメモリからカセットテープにセーブするにはCSAVEコマンドを使います．CSAVEコマンドを使う前にまず，カセットテープを録音したい位置まで進めておいてください．もし，テープの最初から録音するのであればテープの頭出しをする（テープカウンタが5ぐらいになるまで早送りする）必要があります．これは，カセットテープの最初の部分は磁気テープになっていないためです．

G. オーディオカセットレコーダ

次にカセットレコーダを録音状態にしてから,

```
CSAVE "sample" ↵
```

とタイプしてください. テープが回り始め, プログラムがテープに sample というファイル名で記録されます. 記録が終わると, テープの回転は止まります.

リモート端子のない場合, 録音状態にした時点でカセットレコーダが動作を始め, 録音が終了しても動きは止まりません. したがって, 自分で STOP ボタンを押すなどしてテープの動きを管理しなければならないので気をつけてください.

3 プログラムのセーブの確認

プログラムをテープにセーブしても, 必ずしも正しく記録されているとは限りません. テープに傷があったり, 雑音などが入るためです. そのため, テープに正しくセーブされたか否かをチェックする CLOAD? コマンドが用意されています.

まず, テープをプログラムがセーブされているところの少し前まで巻き戻してください. 次に再生ボタンを押して

```
CLOAD? "sample"
```

とタイプしてください. テープが動き始め, sample が見つかると

```
Found : sample
```

と表示されます.

正しくセーブされていると

```
Ok
```

とだけ表示され, テープは止まります. 正しくセーブされていない場合は

```
Verify error
Ok
```

と表示されます. このときは再びセーブし直さなければなりません.

G. オーディオカセットレコーダ

4 プログラムのロード

プログラムをテープからメモリにロードするには，CLOAD コマンドを使います．手順としては，CLOAD？コマンドを使用するときと同じです．つまり次のようになります．

1. テープをセーブされているプログラムの手前にセットする
2. 再生ボタンを押す
3. CLOAD コマンドを入力する

ファイル sample をロードするには，次のように入力します．

```
CLOAD "sample" ↵
```

ファイルが見つかると

```
Found : sample
```

と表示され，ロードし終ると

```
Ok
```

と表示され，テープは止まります．

リモート端子のない場合は，セーブする場合と同様に再生ボタンを押した時点でテープが動き始め，ロードが終了しても，テープは止まりません．したがって，タイミングよくコマンドを入力し，カセットレコーダも自分で止めなければなりません．

正しくロードされたかどうかは LIST コマンドでメモリの内容を見ればわかります．

5 早送りと巻き戻し

リモート端子のついているカセットレコーダの中には，リモート状態では早送りや巻き戻しのできないものがあります．この場合，MOTOR コマンドを実行すると，早送りや巻き戻しができるようになります．

まず

```
MOTOR ON ↵
```

と入力すると，リモート状態が解除されます．この状態で早送りや巻き戻しをして，テープを所定の位置まで動かしてください．そこで

G. オーディオカセットレコーダ

MOTOR OFF

と入力すると，リモート状態が設定されます．ここでCLOAD, CSAVEコマンドを入力してください．

6　カセット使用上の注意点

① ボリュームは高目にセットしてください．あまり低いとノイズの影響を受けやすく，高すぎると飽和（音が割れてしまう）の問題が起こります．

② トーンコントロールは高音（Hi）を多少強めにしてください．高周波数側に情報成分が含まれているからです．

③ セーブとロードは同じカセットレコーダで行うようにしてください．これはテープ走行スピードが微妙に違うことがあるからです．

④ レコーダの中には，再生のときにマイク端子からノイズの入るものがあります．このときは，マイク端子を外して再生してみてください．

⑤ 記録速度2400ボーでのセーブ，ロードは条件が厳しく，一般のカセットレコーダや低級なカセットテープではエラーが発生することがあります．専用のカセットレコーダRQ-8300をご使用になることをお勧めします（特に指定しない時は記録速度は1200ボーです）．

⑥ カセットレコーダは，テレビからできるだけ離してください．

プログラムをカセットテープにセーブするのは，コンピュータがプログラムを音の信号に変換して出力し，カセットテープに録音するということです．従って原則的には音楽を録音するのとまったく同じことです．音楽と違うところは，プログラムはテープの回転ムラや雑音の影響を受けやすいということだけです．

H. カートリッジ

本機の前面にはカートリッジを装着するためのスロットが2つあります．カートリッジには，ROM，RAM，インターフェイスなどの種類があります．

RAMカートリッジ…………………RAMの増設に使用する．

ROMカートリッジ…………………既成のプログラムを，電源ONと共に自動的に実行する．

インターフェイスカートリッジ……インターフェイスを必要とする周辺機器（プリンタなど）を接続するのに使う．

カートリッジの機能はそれぞれ違いますので，詳しい使用法については，カートリッジ付属の取扱説明書を参照してください．

カートリッジを装着するときは次のような点に注意してください．

1. スロットに差し込むときや抜き取るときは本体の電源を必ずOFFにしておく．
2. カートリッジの取扱説明書を読んで，カートリッジを差し込む方向や，左右どちらかのスロットを使うのかを確かめておく．特に指定のないものは，いずれのスロットに差し込んでもかまわない．
3. カートリッジがしっかりと差し込まれていることを確かめてから，本体の電源を入れる．

カートリッジは精密な部品でできたものですから，フタをあけたり，差し込み部分（金属製の端子）を手で触れたり水に濡らしたりすることは避けてください．故障の原因となります．

I. ジョイスティック

ジョイスティックとは

ジョイスティックは図のような形をしたもので，ゲームなどに使うと面白く，遊ぶことができます．

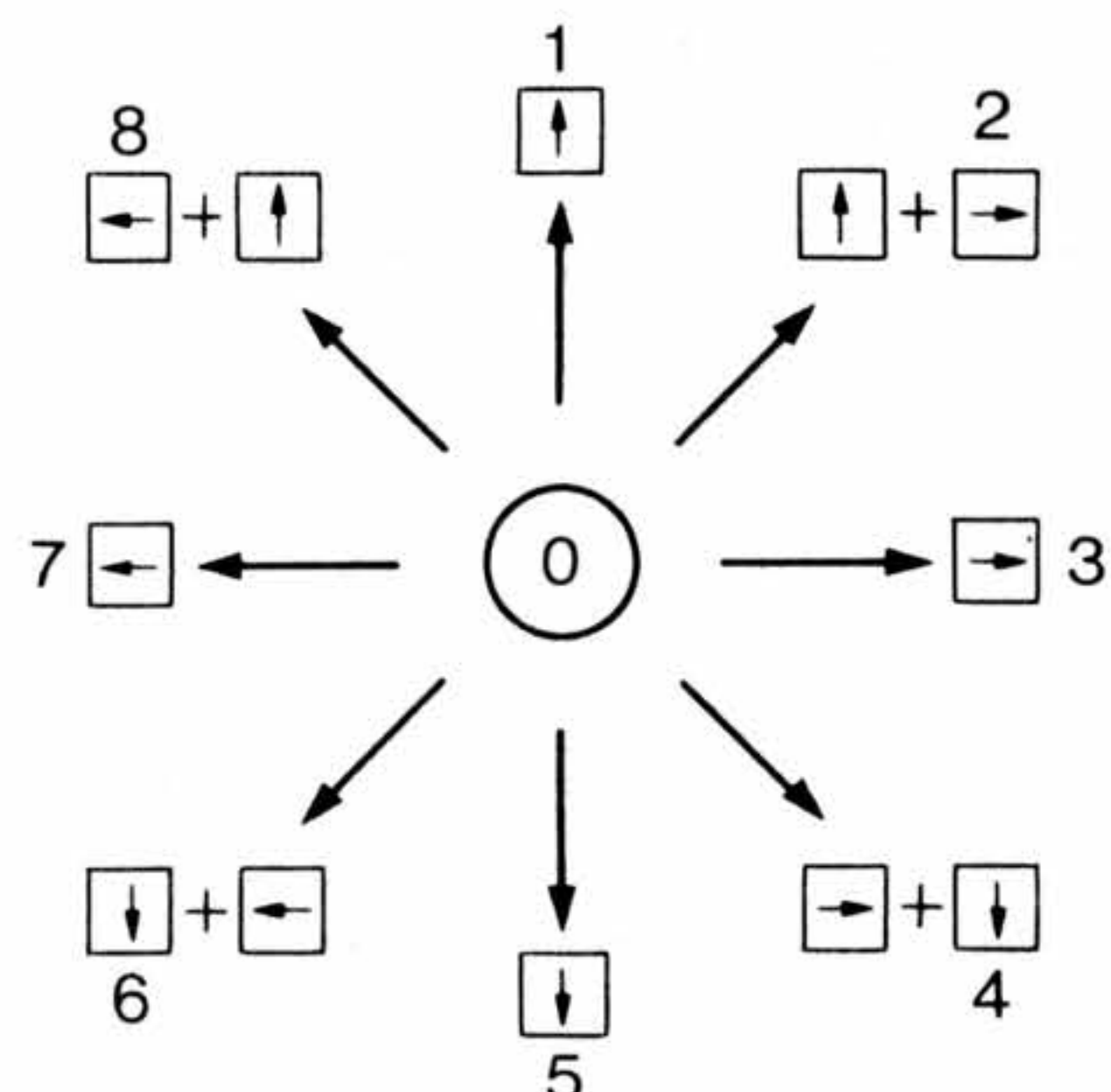
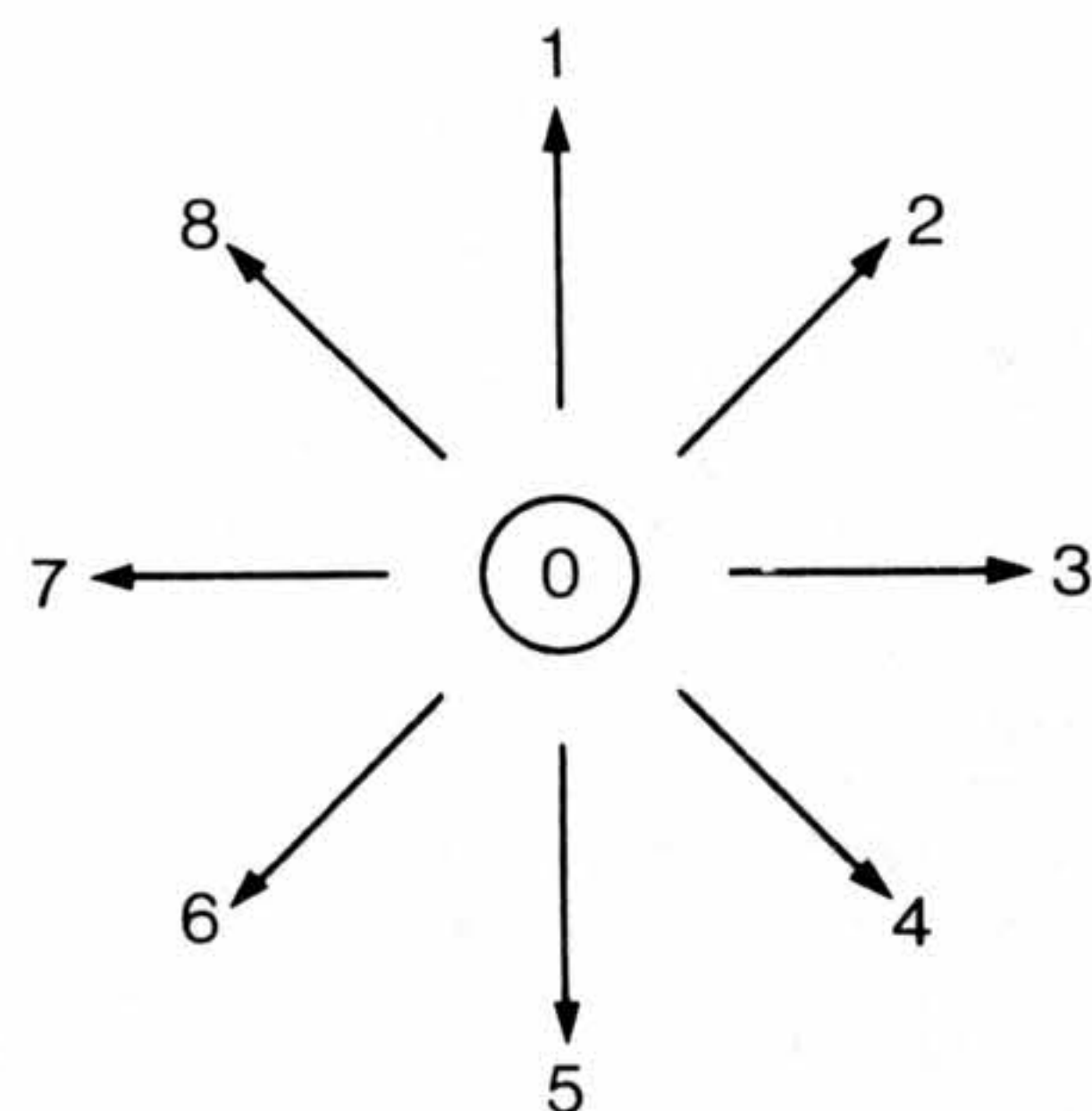
ジョイスティックを使うときは，ジョイスティックのプラグを本体の汎用ポート端子に差し込んでください．最大2つのジョイスティックを接続することができ，それぞれ，ジョイスティック1，ジョイスティック2と呼ばれます．

ジョイスティックの基本的な機能は次の2つです．

- **グリップを8方向に動かす**
- **ボタンを押す**

なお，ジョイスティックがなくても，カーソルキーとスペースキーを使って，ジョイスティックと同等の機能をもたせることができます．

- **カーソルキーを押す**
  …グリップを動かすことに相当
- **スペースキーを押す**
  …ボタンを押すことに相当

このときは，ジョイスティック0として参照します．

J. アフターサービスについて

1 保証書

保証書はご購入日から１年間有効です．必ず「販売店名・購入日」等の記入を確かめて販売店から受取っていただき，内容をよくお読みの上大切に保管してください．

2 修理を依頼されるとき

次の表に従がって調べていただき，直らないときには必ず電源プラグを抜いてから後の処置をしてください．

症　状	原　因	調べるところ
画面が出ない	電源が入っていない	電源または電源コードを調べる．
	接続不完全	接続方法〔(4)ページ〕を見て正しく接続し、プラグやコネクタを充分差し込む．
	テレビのチャンネル違い	RFコンバータのチャンネル切り換えスイッチとテレビのチャンネルを合わせる。ビデオ入力の場合はテレビもビデオ入力にセットする．
色がつかない	カラーテレビの調整不良	テレビの取扱説明書に従って、色、コントラストなどのつまみを調整する．
	プログラム	COLOR文が実行されているか．
文字が読みにくい	テレビの調整不良	同調微調整・カラー調整を行う．
画面が流れる	テレビの調整不良	同調、垂直同期、水平同期を調節する．
でたらめな文字だけが出る	コンピュータの起動不良	電源を一度切って入れ直す．
キーボードがきかない	コンピュータの暴走（プログラムの誤り）	CTRL と STOP キーを同時に押す．それでもだめなら一度電源を切って入れ直す．
CSAVE, CLOAD ができない	カセットレコーダのトラブル	録音・再生ケーブルは正しく接続されているか． カセットの操作〔(20)ページ〕をもう一度確認． 2400ボーでSAVEされたテープは専用データレコーダ以外ではLOADできないことがある．

J. アフターサービスについて

- 保証期間中は

  恐れ入りますが，製品に保証書を添えて，お求めの販売店までご持参ください．
  保証書の規定に従って，販売店で修理致します．

- 保証期間が過ぎているときは

  お求めの販売店にまず相談ください．
  修理すれば使用できる製品については，ご希望により有料で修理致します．

- アフターサービス等について，おわかりにならないときは最寄りの「ご相談窓口」にお問合せください(別紙表参照)．

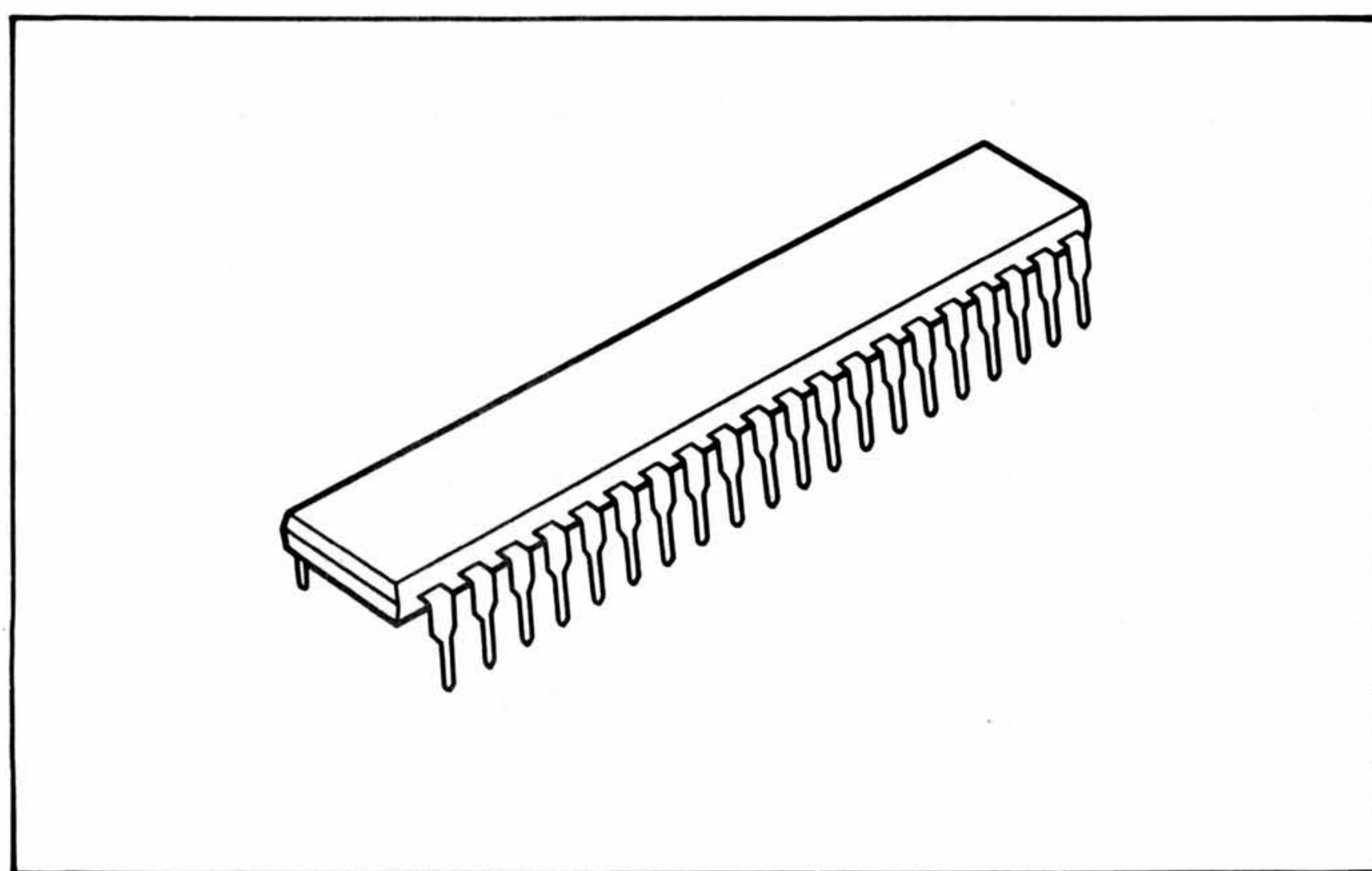
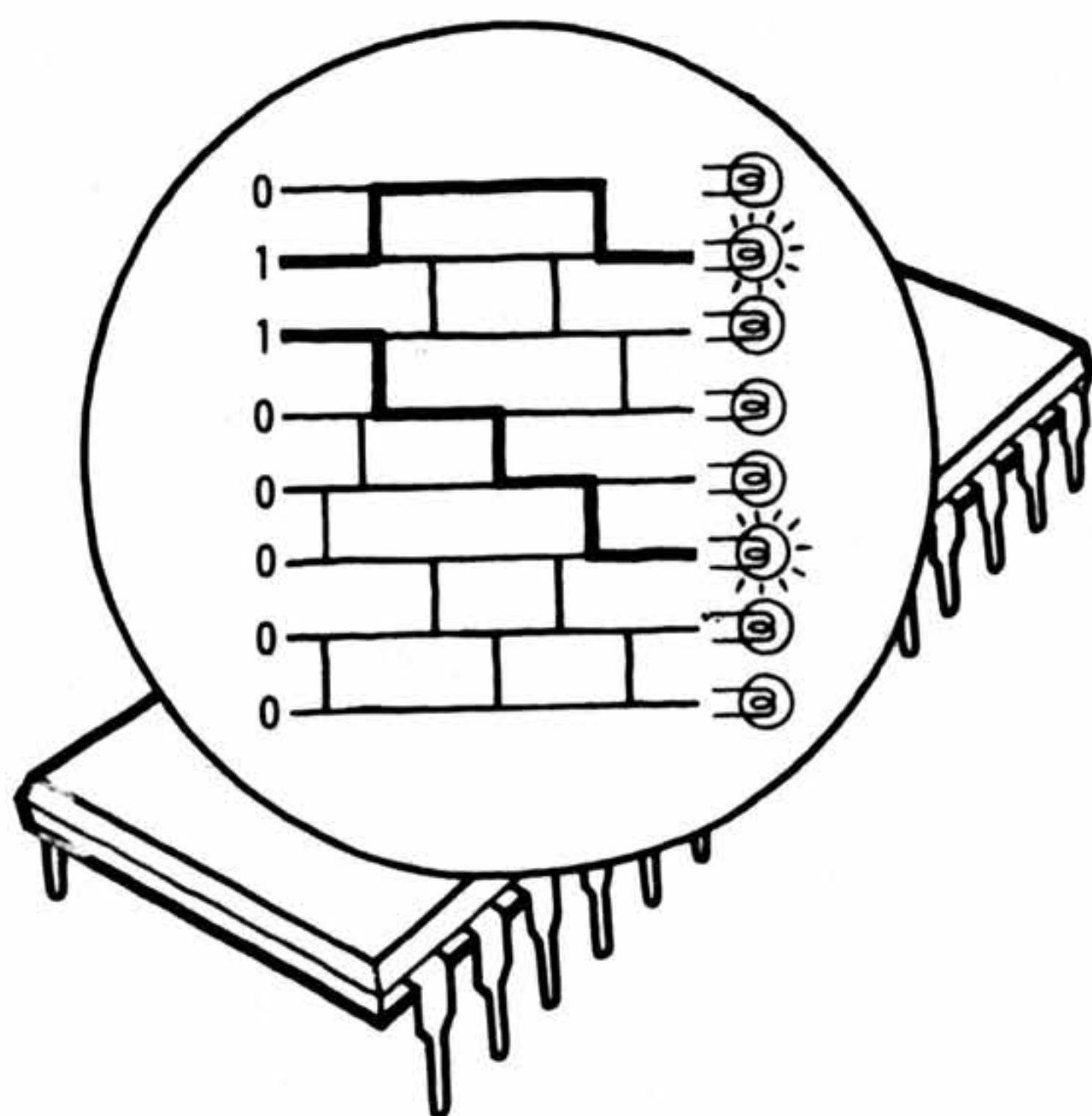
解説編

2

1．コンピュータの正体

「コンピュータは何をやってくれるのか，どんな質問に答えてくれるのか」…SF映画の中などで，わからない質問に悩んだコンピュータが煙を出して壊われてしまうシーンなどを見ていると，そんな考えも浮かびますが，現実のコンピュータではこんなことは起こり得ません．なぜならコンピュータと呼ばれる物は「考える力」をまったくもっていないからです．真空管のコンピュータから現在の第4，第5世代といわれる最新のコンピュータに至るまで計算速度や大きさなどはずいぶん進歩しましたが，「考える力」はあいかわらず0のままです．当分の間は，この力が追加される予定も見込みもありません．それに，もしそんな力をつけてしまったらそれはもうコンピュータと呼ぶべきものではなくなってしまうでしょう．

ではコンピュータとはいったい何なのでしょう．結論から言ってしまえば計算と記憶を専門とする機械です．つまり，ソロバンとメモ帳を一緒にした，電気で動く道具以上の何物でもありません．メモ帳に仕事を書くのはキーボード，ソロバンの答えの表示はテレビモニタというのが一般的ですが，電気信号に変えられるものなら周辺機器を通して音，光，何でも入出力することができます．このソロバンにあたるのがCPUと呼ばれるLSI(大規模集積回路）で，コンピュータとはこのCPUであるといっても過言ではないほどです．MSXではCPUとしてZ80相当のものを使っています．

計算
記憶
周辺機器
CPU
LSI

1. コンピュータの正体

このLSIには多くの足がありますが，この足とその足に電圧をかけるとあの足に電圧がかかるという具合にそれぞれ役割が決めてあります．大変複雑な回路ではありますが，そのひとつひとつの単位は「電気を流せば豆電球が光る」というようなごく単純なものです．

メモ帳の方はメモリと呼ばれ，そこにはいろいろな命令やデータを記憶させることができます．記憶といっても電圧の有無を順番に並べて置くというだけのことです

メモリ
P 102

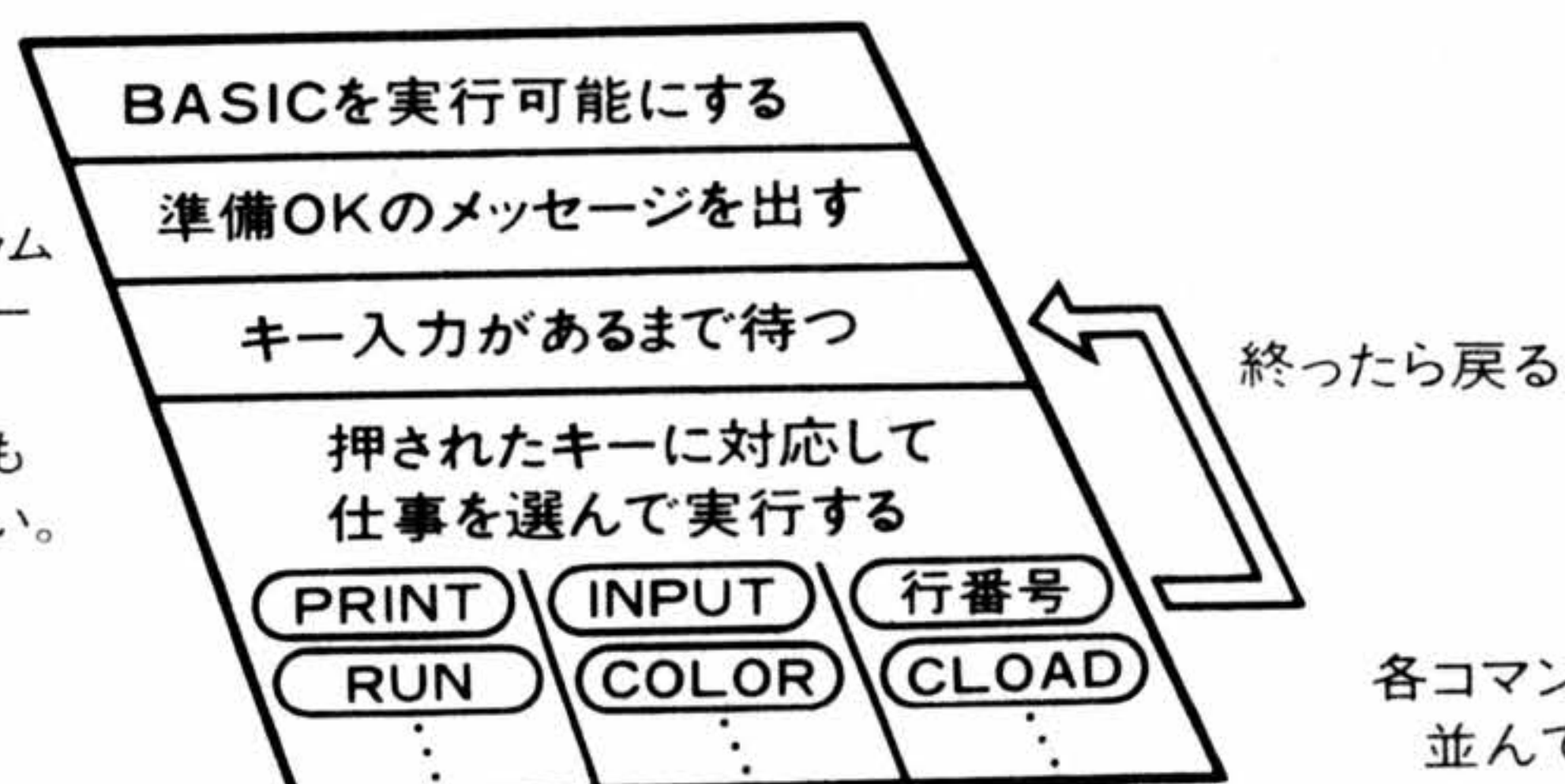
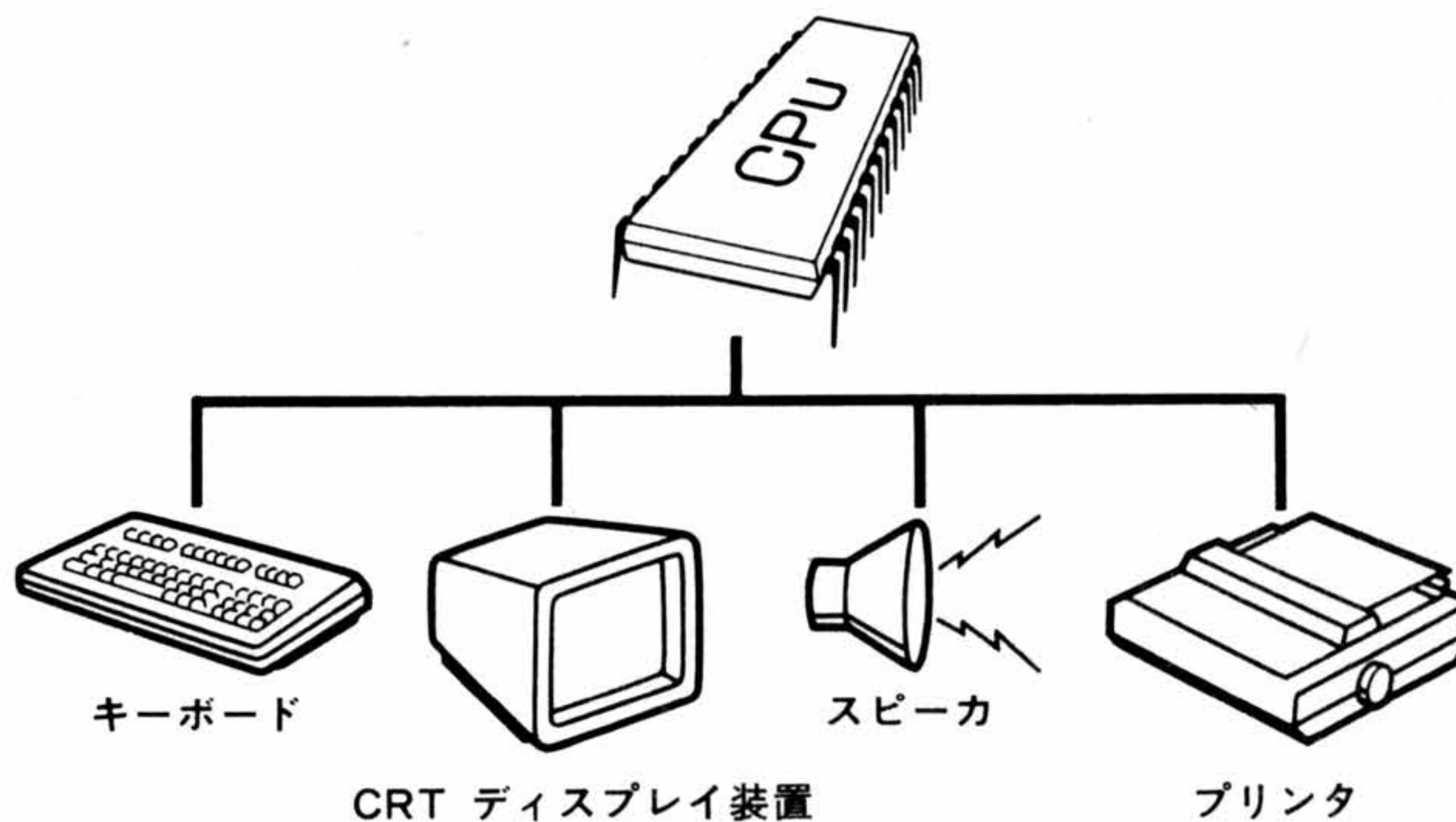
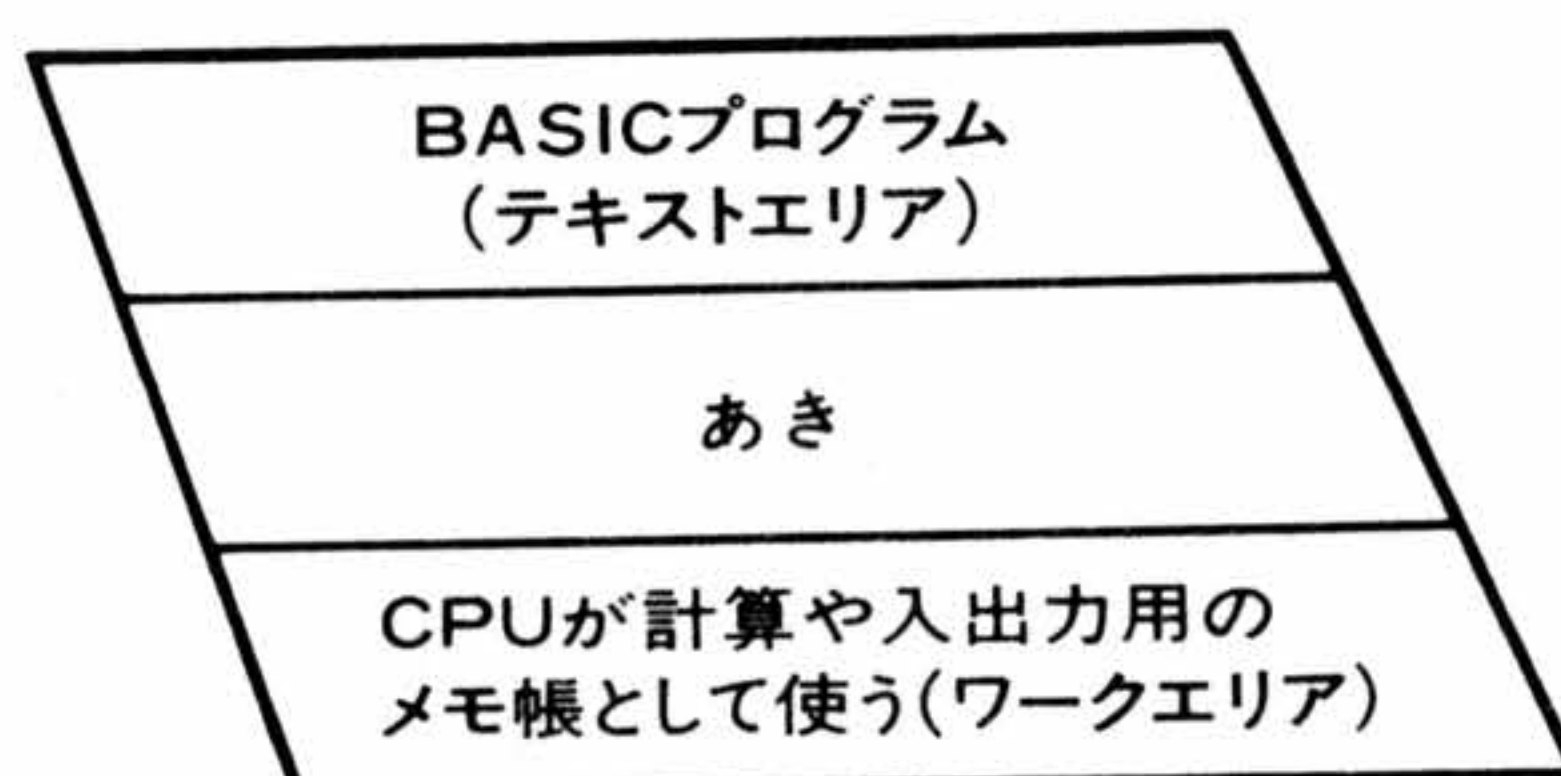
CPUは電源を入れるとまずこのメモリの内容を順番に読むようにできています．一番はじめの方には命令が並んでいなければ何もできないことになります．しかしこの命令は実は非常に種類が限られていて，ソロバンの玉を一つ上げるとか，その値を何番の足にデータとして出すというようなことしかできません．この単純な命令を気の遠くなる程組み合わせてやっと，「キーボードから〝？10＋10″と打つと20と画面に出る」というようなことができるのです．この命令の集りを機械語プログラムと呼びますが，これをメモリに書くのは誰にでもできることではありませんし，たった1回の足し算のためにいちいち何百何千というステップの命令を組み合わせるのではたまりません．

機械語
P 106

ステップ

BASIC

そこでMSXでは，メモリのはじめの方にBASICと名づけられた言語(約束事）に従ってCPUが動くように機械語でプログラムが書いてあります．？というキーが押されたら，それは画面に出力するための準備だとか，A＝3は変数Aを用意して3を代入することだとか決めておくわけです．これではじめてBASICという「少しは人間にもわかりやすい」命令をCPUに伝えることができます．

1．コンピュータの正体

このようにBASICを機械語に翻訳してくれるプログラム（BASICインタープリタ）は，ROMというメモリに書いておきます．ROMの内容は，いってみれば石板に刻んだ文字のようなもので一度工場で作られたが最後，電源を切ろうが，その内容を消せと命令しようが決して消えません．つまりキーボードをどのように操作しても，MSXの心臓部であるBASICインタープリタを壊すことはできないのです．

インタープリタ

ROM
P 103

RAM
P 104

これに対してRAMというメモリは，消したり書いたりできる黒板のようなもので，電源を切るとその内容は消えてしまいます．しかし大きな数の計算や，その為のBASICプログラムは使う人や目的によっていろいろと違いますから，それを置くためにRAMはなくてはなりません．

たとえば「数字と記号のキーが押されたら，それに従って加減乗除を行って答を表示する」というような決まった仕事しか必要ないコンピュータ（電卓）なら計算用にほんの少しのRAMが必要なだけで，仕事は全部ROMに書き込んでおけばよいことになります．

2．プログラムの作成と実行

MSX のコンピュータの電源を入れると，画面にはメッセージが現れます．〝Ok″の表示後カーソルが現れてすべてのキー入力を受け付け，それらは画面に表示され，カーソルが進みます．この状態をダイレクトモード，またはコマンドレベルと呼びますが，このモード中は2つの作業が可能です．

ダイレクトモード

コマンドレベル

1つは何かのコマンドを与えて，すぐに実行させることです．たとえば，次のようにします．

```
?10*10
 100
Ok
PLAY "ABC"
Ok
A=5:BEEP
Ok
□
```

これらのコマンドは，RETURN キーが押されると同時にコンピュータに伝わり，実行されてまたキー入力待ちになります．このとき「コマンドを実行しました」という意味で〝Ok″が表示されます．

もう1つは行の先頭で数字を入力し，続けて何か書くことによってプログラム行を作ることです．

プログラム行

```
10  INPUT  A
20  PRINT  A*2
□
```

このときも RETURN キーを押すとはじめて「この行をプログラムとして覚えておけ」という人間の命令がコンピュータに伝わります．人間が入力したものの先頭にまず数字があったらそれはプログラムの行だとコンピュータが解釈するようになっているのです．このときは一行を覚えるだけで〝Ok″は表示しません．

2．プログラムの作成と実行

こうしてプログラムができたら，RUN コマンドを入力してやるとプログラムの実行が始まります．

実行

```
RUN
? □
```

プログラムは現在10行の INPUT 命令を実行中です．何か値を入力してやらないと，コンピュータはいつまでも待っています．5 RETURN と入力してみましょう．

```
?  5
 10
Ok
□
```

これで20行の PRINT A ＊ 2 が実行され，その次にはプログラムの行がないのでプログラムの実行は終了し，また〝Ok〞が表示されています．

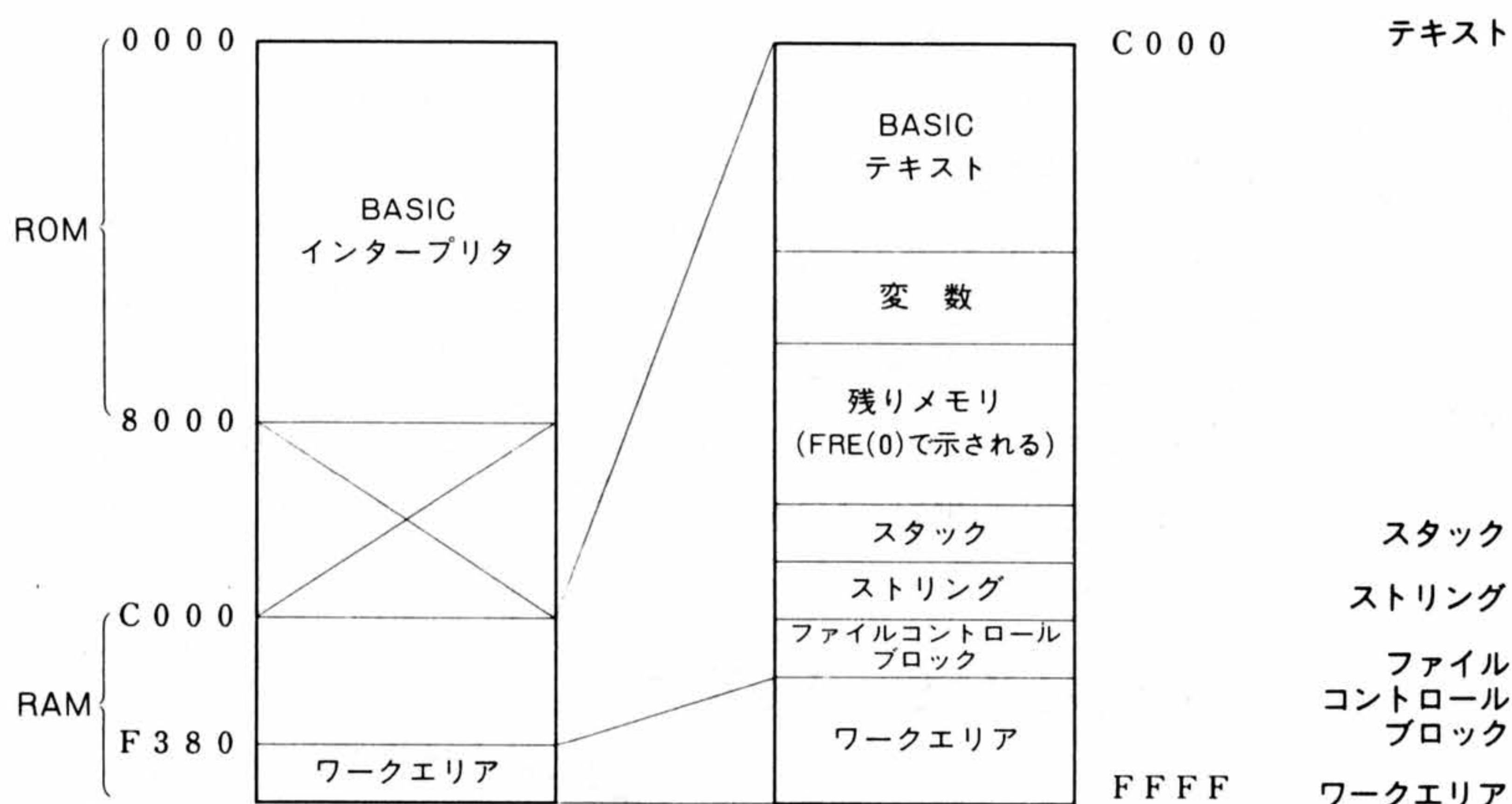
ここまでの動作は，すべて入門編で詳しく解説してありますが，さて実際にこれらはどのように記憶され，実行されているのでしょう．

コンピュータの記憶については「メモリ」の項で解説してありますが大体次ページの図のようになっています．

この図の中で，「BASIC テキスト」から「ストリング」までが BASIC で自由に使える領域です．MSX の電源投入直後は，この中には 0 か255という値がはいっていて，プログラムや変数は何もないことになっています．このとき FRE（0）の値は12431になります．

次に，10 INPUT Aや20 PRINT A ＊ 2 というプログラムを作るとテキストエリアにこれらが格納されます．PEEK 関数で&HC000 から20バイトほどメモリの内容を調べてみると

```
0，9，192，10
```

2．プログラムの作成と実行

のように値がはいっています．実はこれがBASICプログラムの本体(テキスト）で，LISTで現れる

```
10 INPUT A
20 PRINT A * 2
```

とはずいぶん違っているように見えます．しかしROM内のBASICインタープリタは，これらをデータとして動作します．ダイレクトモード中はインタープリタはワークエリアの方に注目するようにできていて，人間がいろいろとキーを押している間にはそれをワークエリアに書きながら画面にも出力し，RETURNが押されると今入力されたものはコマンドか，プログラム行かを判定します．コマンドならそれを実行し，プログラム行ならこのテキストエリアに書き込んでいくのです．

中間コード

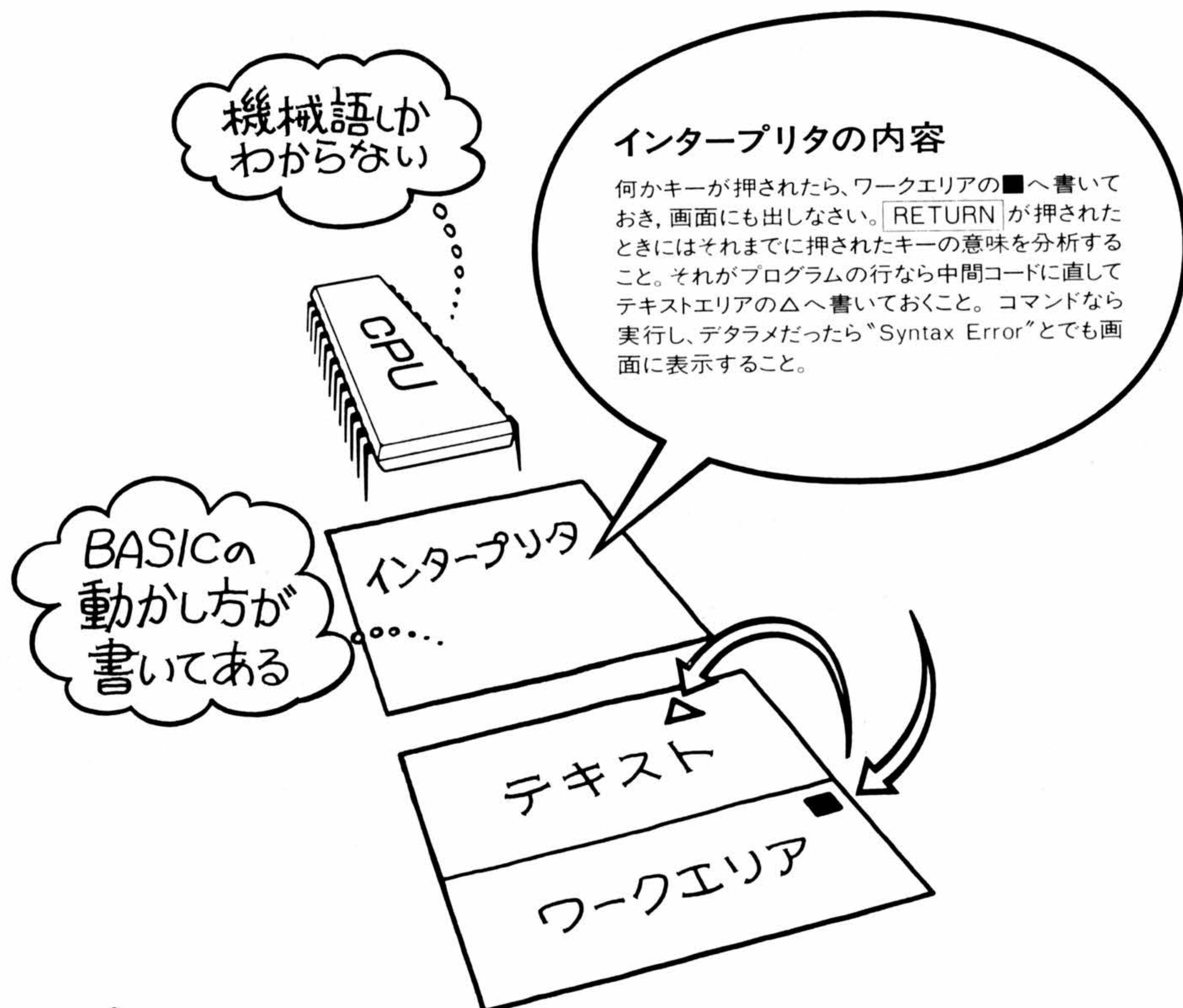
こうしてでき上ったテキストは，中間コード，行番号，その他でできています．中間コードとはINPUTなら133, PRINTなら145のように決めたコードのことです．コンピュータが「INPUT」という決まった動作をするのに，いちいち5文字を読むのでは遅くなってしまうし，メモリも余計に使ってしまうからです．

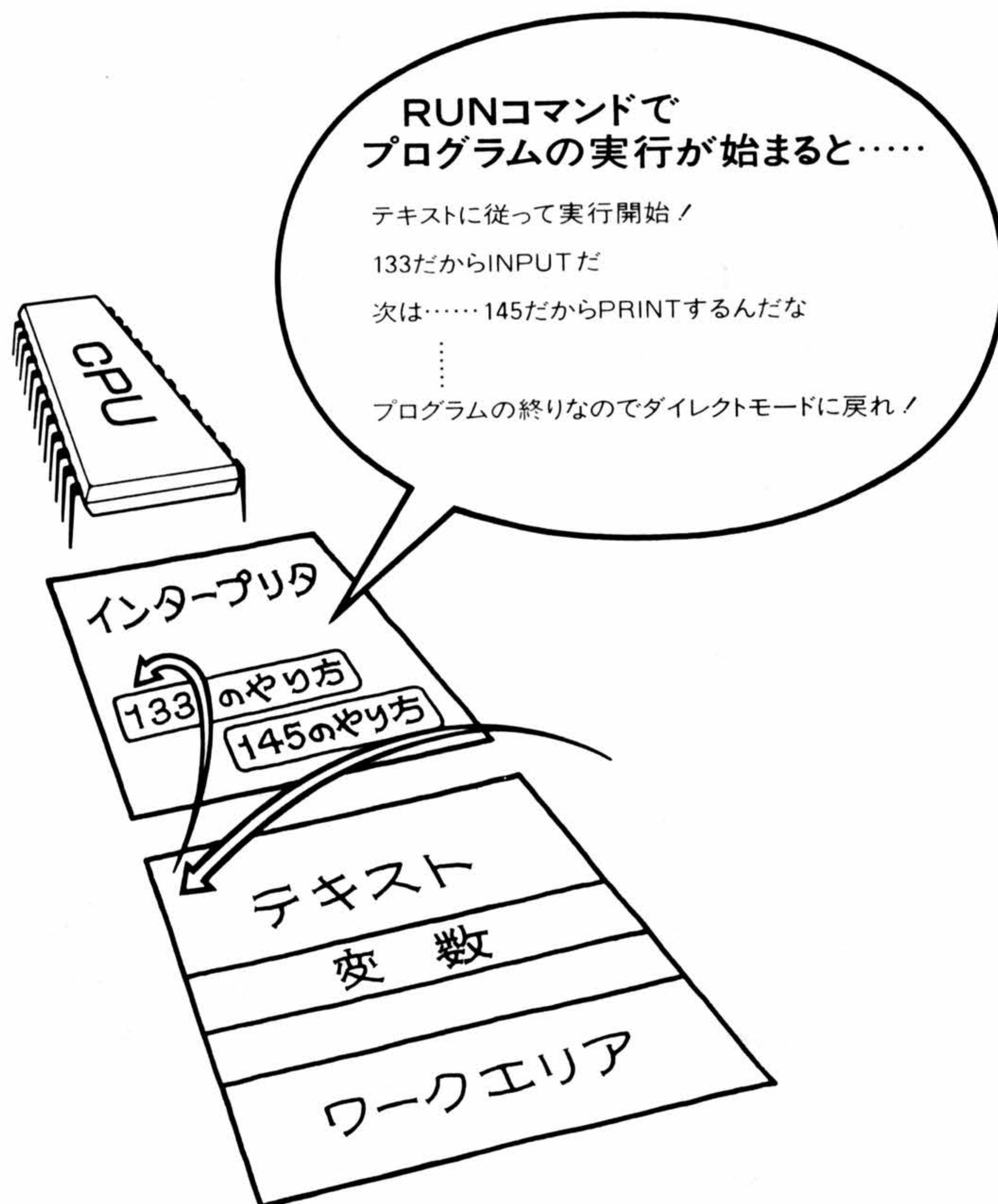
2．プログラムの作成と実行

いよいよ RUN コマンドが入力されると，インタープリタがテキストを先頭から見ながら実行し，変数が必要になれば変数領域（テキストの後に続いている）に用意します．そして END（または STOP）というステートメントにつきあたるか，テキストの一番後に行ってしまって，その先がなければ実行を終わってまたダイレクトモードになるという訳なのです．もちろんプログラムの実行中に CTRL ＋ STOP キーが押されるとか，エラーが発生した場合も実行は中止してダイレクトモードになります．

エラー P112

最後にこれらを図解しておきますが，「コンピュータの正体」や「メモリ」の項も参考にしてください．

●ダイレクトモードの場合

2．プログラムの作成と実行

●プログラム実行中

3. モード

コンピュータに命令を実行させるのに2つのモードがあります．1つは命令を入力して，RETURN キーを押すとすぐ実行する方式でダイレクトモードと呼ばれます．もう1つは，命令を入力して，RETURN キーを押しても，そのときは実行されずに，メモリに格納され，あとからまとめて実行される方式でプログラムモードと呼ばれます．

ダイレクトモード

プログラムモード

● ダイレクトモード

通常，BASICの命令を入力して，RETURN キーを押すと，命令はすぐ実行されます．たとえば

PRINT "HELLO"

とキー入力して，RETURN キーを押せば，すぐPRINT命令が実行され，

HELLO

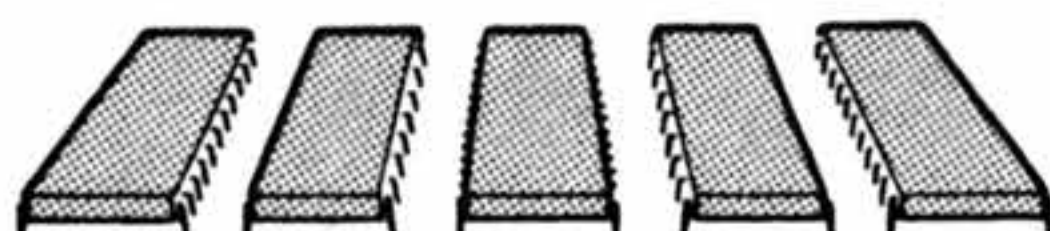
と画面に表示されます．

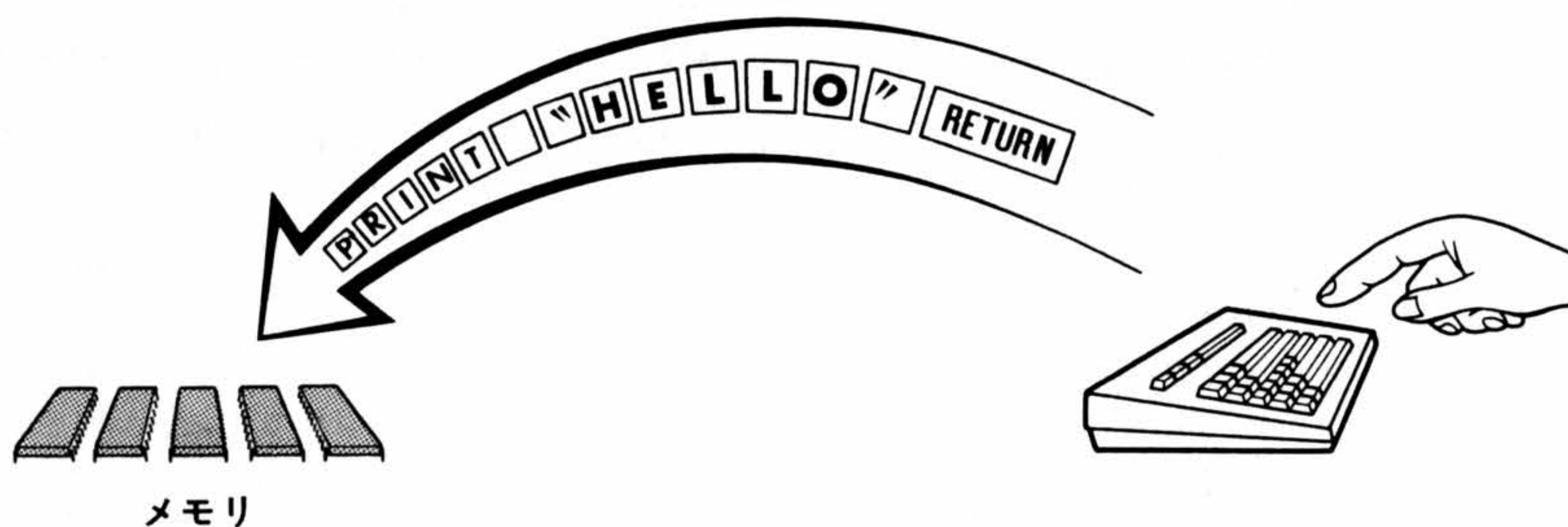
反対にどんなに正確にキーボードから命令を入力しても，RETURN キーを押さなければ命令は実行されません．

バッファ

というのは，キーボードからタイプした文字はキーボード用のバッファに蓄えられ，コンピュータには伝わらないので，コンピュータはどんな命令が入力されているのかわからないからです．RETURN キーを押すと，はじめて，キーボードのバッファの内容がコンピュータに転送され，コンピュータは入力された内容を理解できるようになるのです．

PRINT "HELLO"

キーボードバッファ

メモリ

3．モード

コンピュータに送られた命令は，すぐ実行され，実行が終わるとメモリから命令が消えてしまいます．そのため，同じ命令を実行させるためには，再び命令をキー入力しなければなりません．

ダイレクトモードの使い方としては，たとえば$10\times5^3+6-30\div5$を計算しようと思ったら

```
?10*5^3+6-30/5
```

とキー入力して，最後に RETURN キーを押せば，入力行のすぐ下に計算結果が

```
1250
```

と表示されます．

ダイレクトモードはこのように簡単な計算をしたり，「実行せよ」「リストをとれ」「メモリをクリアせよ」といったような簡単な指令をコンピュータに与えるときに便利です．しかし，複雑な処理を行わせようとするとダイレクトモードではとても間に合いません．このときはプログラムモードを使って，プログラムを組まなければなりません．

3. モード

● プログラムモード

命令の先頭に行番号を付けると, RETURN キーを押してもメモリに転送されるだけで実行はされません. 行番号のついた命令はメモリ内にいくつも格納することができます. そして, メモリ内での命令の順番は, キー入力した順ではなく, 行番号の若い順です. たとえば

```
30 NEXT I
20 S=S+I
40 PRINT S
10 FOR I=1 TO 10
```

の順にキー入力しても, メモリ内では次の順に並んでいます.

```
10 FOR I=1 TO 10
20 S=S+I
30 NEXT I
40 PRINT S
```

このようなメモリに格納されている命令群をプログラムと呼びます. プログラムはRUNコマンドによって実行されます. 実行してもメモリから消えてしまうことはありません. したがって何度でも, 実行させることができます.

● 行と行番号

行

プログラムでは, 行番号から始まって, RETURN キーを押すまでを行と呼んでいます. 行はこの行番号と命令（ステートメント）からなります.

```
10       N=20
行番号   ステートメント
```

上の行を指すときは行番号をつけて「10行」といった呼び方をします.

行番号は, その行がプログラムの一部であること（RETURN キーを押しても実行せずにメモリに格納する）を示すとともに, RUNコマンドで実行するときの実行の順番を示しています.

プログラムの1行の長さは最大255文字までです. 画面に表示される文字の1行 とは意味が違うので注意してください.

4．プログラムのロード/セーブ

メモリ内のプログラムは電源スイッチをOFFにすると消えてしまいます．再び，同じプログラムを実行するには，またプログラムを作り直さなければならないのでしょうか．

外部記憶媒体

その心配はありません．電源を切る前にメモリ内のプログラムを他の外部記憶媒体に転送記憶しておき，再び実行させるときは，その外部記憶媒体からメモリにプログラムを転送すればよいのです．このように外部記憶媒体に保存することをセーブすると言い，外部記憶媒体からメモリに転送することをロードするといいます．

セーブ

ロード

ディスク

外部記憶媒体としてよく使われるのはカセットテープとディスクです．カセットテープやディスクを使ってプログラムのロード／セーブをするときは，それぞれ次のようなコマンドを使います．

カセットテープにセーブ……CSAVE
カセットテープからロード……CLOAD
ディスクなどにセーブ……SAVE
ディスクなどからロード……LOAD

カセットテープにプログラムをロード／セーブする方法については入門編で述べましたので，ここでは触れません．詳しくは入門編を参照してください．ここではディスクにロード／セーブする方法について簡単に触れておきます．

ここでの説明はすべて，実際にディスク装置とシステムディスクが使用されているときを考えているものです．ディスク装置を使用していないときはSAVE，LOADの各コマンドはカセットテープ上でアスキー型式のファイルを扱うために使われます．

ディスクは円盤状の記憶媒体でカセットに比べて次のような特徴があります．

- 高速である
- 大容量である
- 操作し易い
- 信頼性が高い
- 高価である

4. プログラムのロード/セーブ

カセットテープにプログラムを記憶するのにカセットレコーダが必要なように，ディスクに記憶するにもディスク装置が必要です．このディスク装置には1枚のディスクしか入らない1ドライブ方式のもの，2枚のディスクが入る2ドライブ方式のものなどがあります．またディスクには3インチ，5¼インチ，8インチなどの標準フロッピィディスクがあります．

フォーマット

まず，ディスクをディスク装置に入れます．もし，このディスクがフォーマットされていないディスクであれば，最初にフォーマットをする必要があります．フォーマットが済んでいれば次のようにタイプしてください．

SAVE "XYZ"

ファイル P96

メモリ中のプログラムがディスクにセーブされ，XYZというファイルが作られます．ただし，これはディスクがドライブ1に入っている場合です．ドライブ2にあるときは

SAVE "2: XYZ"

とします．

プログラムをディスクからメモリにロードするときは

LOAD "XYZ" または LOAD "2: XYZ"

としてください．

どうですか．簡単でしょう．カセットの場合のように録音ボタンを押すとか，早送りするとかいった面倒な操作は一切不要なのです．しかも，カセットよりはるかに高速で，どんなに長いプログラムでも，あっというまにロード／セーブを完了してしまいます．また，ロード／セーブが失敗するということもほとんどありません．そのため，CLOAD ? に該当するような命令はディスクにはありません．

5. 命令

コンピュータに対する「ああしろ」「こうしろ」といった指令のことを命令と呼んでいます．たとえば

```
GO TO 10
```

は「行番号10へ行け」と指令する命令です．

この命令の中で，ダイレクトモードで使われるものをコマンド，プログラムモードで使われるものをステートメント（文）と呼んで区別しています．

コマンド

ステートメント

たとえば，プログラムを実行するための命令

```
RUN
```

は通常ダイレクトモードで使われるので，コマンドです．一方，キーボードからデータを入力するための命令 INPUT は通常

```
50 INPUT X
```

のように，プログラム中で使われるのでステートメントです．しかし，実際には，ダイレクトモードとプログラムモードの両方で使える命令が多いので，コマンドとステートメントを厳密に区別するのは難しいことです．

MSX BASIC で使う命令はすべて英単語を基礎としているので，アルファベットだけで表現されます．これらの命令と後述する「関数」に使われる単語を合わせて，予約語と呼んでいます．

予約語

予約語は，大文字でも小文字でもまったく同じものとして扱われます．たとえば run，RUn，rUn などは RUN と同じです．また，小文字で

```
10 goto 10
```

のようなプログラムを入力しても LIST を取ってみるとすべて大文字にされていることがわかります．

```
LIST
10 GOTO 10
Ok
```

5．命令

● コマンド

ダイレクトモードでよく使われるのは次のような命令です．

RUN ……… プログラムを実行する

LIST ………プログラムのリストをとる

NEW ……… プログラムを消す

DELETE…… プログラムの行を消す

CLOAD …… プログラムをカセットテープからメモリにロードする

CSAVE …… メモリ上のプログラムをカセットテープにセーブする

RENUM…… プログラムの行番号を変える

このように，プログラムに対して何らかの操作を加えるような命令はコマンドと考えればよいでしょう．

● ステートメント

ステートメントは主にプログラムの中で使われる命令です．次のプログラムを見てください．

```
10 REM **READ***
20 CLS: LOCATE 8, 3
30 FOR I=1 TO 4
40 READ R$
50 PRINT R$;
60 NEXT
70 END
80 DATA Please, read, this, manual
```

このプログラムで使われているREM, CLS, FOR, NEXT, END, DATAなどの命令はすべてステートメントと言えます．また，20行のように1行に複数のステートメントがコロン（：）で区切られて並んでいる形式をマルチステートメントと呼んでいます．

マルチステートメント

6．関数

関数はデータを与えると，それに対して何らかの計算や操作を行い，その結果を返すものです．たとえば，SQR は与えられたデータの平方根を返す関数です．

SQR（100）

は100の平方根を計算し，結果10を返します．またこの与えるデータのことを引数と呼び，関数の一般形は次のようになります．

引数

関数名（引数1，引数2，・・・）

関数は単独では用いられず，必ず命令の中で使用されます．たとえば，単に

SQR（100）↵

としても，エラーになります．この結果を知りたければ PRINT 命令などと組み合わせて

PRINT SQR（100）↵

のようにしなければなりません．

関数には，すでに BASIC の中に組み込まれていて，ユーザーが自由に使えるものがあります．たとえば次のような関数です．

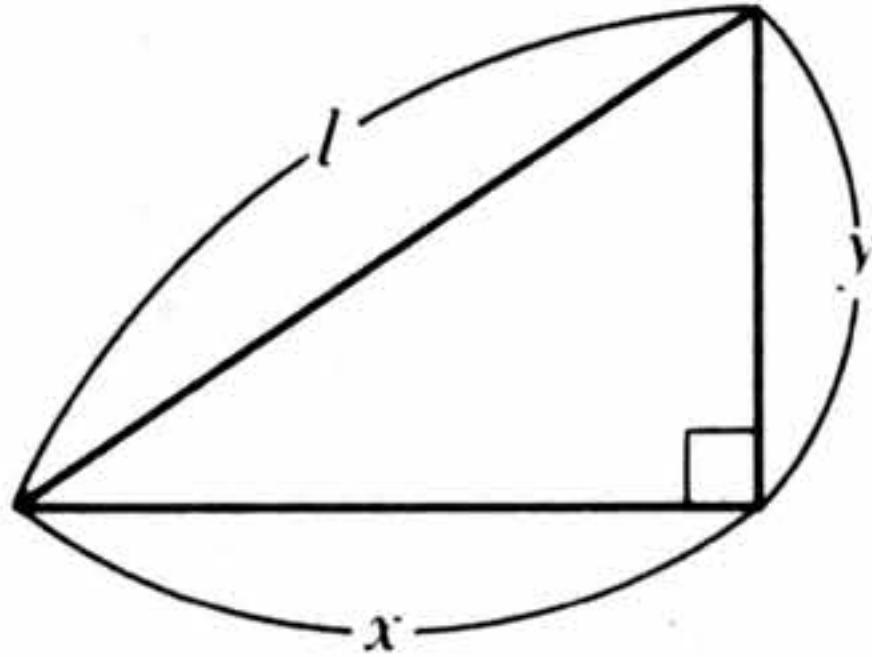
SIN(X)	SIN X を計算する
COS (X)	COS X を計算する
PEEK (I)	I 番地のデータをとり出す
HEX$ (X)	10進 X を16進に変換する

番地 P 102
16進 P 89

このような関数を組み込み関数と呼んでいます．

組み込み関数

また，関数をユーザーが定義することもできます．それには DEF FN 命令を使います．たとえば，次ページの図のような直角三角形の2辺の長さ x，y が与えられたとき，斜辺 l の長さを返す関数 L を定義してみましょう．

6．関数

DEF FNL (X, Y) =SQR (X∧2+Y∧2)

ここで，FNLが関数名となります．

次のプログラムは，x，yの値を，キーボードから入力して，lを計算し，結果を表示するものです．

```
10 DEF FNL (X, Y)=SQR (X∧2+Y∧2)
20 INPUT X, Y
30 PRINT "X, Y, L", X, Y, FNL (X, Y)
40 GOTO 20
```

ユーザー定義関数

このFNLのような関数をユーザー定義関数と呼んでいます．

この他USR関数というものがありますが，これは機械語でプログラムを作らなければなりません．「機械語」の項を参照してください．

7．式と演算

● 式

式とは定数や変数を演算子で結合した数式のことを言います．また，単に数値や文字，あるいは変数1つだけのものでも式ということができます．

```
10＊5−2
10＋3／2
6＋（5＋2）＊3
"MSX"
50
X
```

● 演算

MSX BASICでは演算は次の5つに分類されます．

演算

1. 算術演算
2. 関係演算
3. 論理演算
4. 関数
5. 文字列演算

文字列演算
P 28

このうち，関数と文字列演算については別の項を参照してください．

算術演算　算術演算とは加減乗除やべき乗計算のことです．演算の規則などは，私たちが学校で習ったものと同じですが演算子が多少異なります．

算術演算子には次のものを使います．

演算子

優先順	演算子	演　算	例	意味
①	^	指数演算	5 ^ 3	5^3
②	−	負号	−5	−5
③	＊	乗算	5 ＊ 3	5 × 3
③	／	除算	6 ／ 3	6 ÷ 3
④	＋	加算	9 ＋ 3	9 ＋ 3
④	−	減算	9 − 3	9 − 3

指数演算と乗除算の演算子が普段使っているものと違っているのがわかると思います．これ以外にかっこも使うことができます．

7．式と演算

```
10＋5＊（6＋3）／2
```

／を使った除算は，小数点以下まで商を求めます．たとえば

```
?16／5
 3.2
```

となります．一方私たちが学校で最初に習った除算は整数の商と余りを求めるものでした．さきほどの計算ならば

16÷5＝3…1

と計算されます．このような除算もBASICを使って行うことができます．それには除算演算子を／でなく，¥とします．たとえば

```
?16¥5
 3
```

となります．このとき小数点以下は切り捨てられます．扱われる数値が実数の場合は，演算が実行される前に小数点以下が切り捨てられます．

```
?23.75¥5    （＝23÷5）
 4
```

整数除算の余りは，別に計算しなければなりません．そのときは，演算子MODを使います．たとえば16÷5の余りは次の式で求められます．

```
?16 MOD 5
 1
```

扱われる数値が実数の場合は，演算が実行される前に小数点以下が切り捨てられます．

```
?23.75 MOD 5
 3
```

0で除算を行ったときは，〝Division by zero〟エラーがでて，プログラムは終了します．

オーバーフロー

また，次のような計算をした場合，結果がオーバーフローするので〝Overflow〟エラーが表示されます．

```
A%=1000*10000
B#=2^1000
```

7．式と演算

関係演算

関係演算子は2つの数値を比較するときに用います．結果は，真（－1），偽（0）のいずれかです．主にIF文でプログラムの流れを変えるときに用いられます．

使用される関係演算子は右の通りです．

演算子	意　味	例
=	等しい	A=B
<>	等しくない	A<>B
<	小さい	A<B
>	大きい	A>B
<=	小さいか等しい	A<=B
>=	大きいか等しい	A>=B

もし，関係演算子が算術演算子と共に使用された場合は，算術演算子が先に評価されます．

10=10	真（－1）
10>5	真（－1）
5>10	偽（0）
10<>10	偽（0）

10+5*2=10*3+5

⇩

20=35　偽（0）

関係演算子は次のように，IF文の中でよく使われます．たとえば

```
IF A=B THEN 50
IF I MOD J< >0 THEN K=K+1
```

また，関係演算子を次のように代入文の中で使うこともできます．

```
X=（5=5）            (X=-1)
Y=（5<3）            (Y=0)
Z=（5*3<>15）        (Z=0)
```

7．式と演算

論理演算

ビット操作
ブール演算

論理演算子は複数の条件を調べたり，ビット操作やブール演算を行ったりするのに用います．

論理演算子として次のものがあります．

NOT，AND，OR，XOR，IMP，EQV

NOT（not　否定）

X	NOT X
1	0
0	1

AND（and　論理積）

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

OR（or　論理和）

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

XOR（exclusive or　排他的論理和）

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

7．式と演算

IMP（implication　包含）

X	Y	X　IMP　Y
1	1	1
1	0	0
0	1	1
0	0	1

EQV（equivalence　同値）

X	Y	X　EQV　Y
1	1	1
1	0	0
0	1	0
0	0	1

論理演算子はIF文の中で複数の条件を判定する場合によく使われます．たとえば

```
IF X<0 OR X>99 THEN 100
```

これはX＜0またはX＞99のとき100行へ行けと命令しています．

その他，数値の論理演算を行うときも使えますが，これについては論理演算の項を参照してください．

8．定数

文字型
数値型

精度

定数とは定まった数つまり「変わらない数」のことです．この定数にはいくつかの型（タイプ）があります．大きくわけて，文字型と数値型です．

文字型定数とは文字の前後をダブルクォーテーション（″）で囲ったものです．たとえば

〝HELLO″，〝BASIC″，〝123″

などです．

数値型定数は正，負または0の数の事です．この数値型定数はさらに，整数型と実数型に分けられます．整数型は－32768から32767の整数を表すとき用います．一方実数型は整数および小数を表すとき用います．実数型はさらにその精度によって単精度と倍精度にわけられます．

実数を表すのに，2通りの方法があります．1つはベキ乗の記号を付けて表す方法であり①，もう1つは整数または小数点をつけて表す方法です②．

5E＋5 (5×10^{5})
6.12E－2 (6.12×10^{-2}) ①

500000
0.0612 ②

①のような形式の定数を浮動小数点定数，②のような形式の定数を固定小数点定数と呼んでいます．浮動小数点定数は 10^{-66}～10^{62} の範囲の値を表すことができます．

少し頭が混乱していると思いますので，ここでまとめてみましょう．次のようになります

8．定数

● 整数型

−32768から32767の範囲で，後に型宣言文字%が付いた整数は整数型数値として扱われます．たとえば

31768%
0%
−1689%

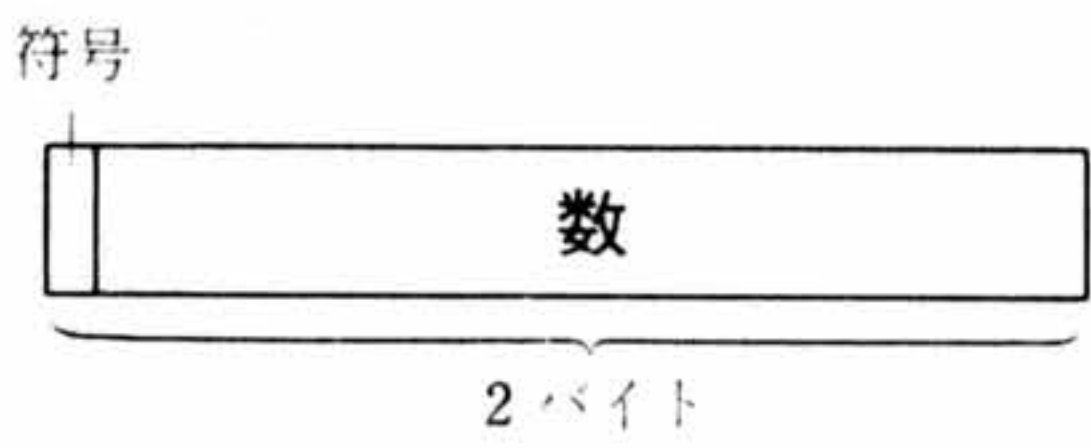
などは，すべて整数型です．ただし%を除いて

31768
0
−1679

とすると倍精度型数値として扱われるので注意してください．

符号ビット

整数型数値はメモリ内では2バイトを占め，左端（MSB）は符号ビットとして使われています．

〜進数
P 89

整数型数値は今まで述べた10進数だけでなく，16進数，8進数そして2進数でも表現することができます．この場合，数値の先頭に次のような記号を付けます．

16進数　&H
8進数　&O
2進数　&B

次にいくつかの例を示しましょう．

&H 76=118　16進
&HFFFF=−1　16進
&O347=231　8進
&B 01110110=118　2進
&B 11100111=231　2進

8．定数

整数型では2バイトの符号ビット（一番上位のビット，MSB）が1だと値が負になりますので，その点を気をつけてください．

&H 8000≦16進数≦&H 7 FFF
&O 100000≦ 8 進数≦&O 77777
&B 1000000000000000≦ 2 進数≦&B 0111111111111111

上の値の範囲を越える数値を指定するとオーバーフローとなります．

● 単精度型

型宣言文字！が後についた実数は単精度型として扱われます．たとえば

100！
1.986！
0.0105！

などです．また，ベキ乗記号としてEを使った浮動小数点定数も単精度型として扱われます．

1E02＝(1×10^{2})
1.986E01＝(1.986×10^{1})
−1.05E−02＝(-1.05×10^{-2})

有効桁数

単精度型数値はメモリ内では4バイトを占め，有効桁数は6桁です．

12345666！→123457
100.5263！→100.526

8．定数

● 倍精度型

型宣言文字のない実数あるいは型宣言文字#が付いた実数は倍精度型として扱われます．たとえば

1234
3.1415956
36 #

などです．また，ベキ乗記号としてDを使った浮動小数点定数も倍精度型として扱われます．

−1.09432 D −06 ＝ (-1.09432×10^{-6})
0.3141592653 D 01 ＝ $(0.3141592653\times10^{1})$

倍精度型数値はメモリ内では８バイトを占め，有効桁数は14桁です．

9．文字列(ストリング)

文字列とは255文字以下のダブルクォーテーション(″)で囲まれた英数字，カナ記号などの列のことです．

● 文字列の加算

数値と同じように，文字列も＋記号を使って加算することができます．ただし，加算の意味はだいぶ違います．文字列の場合，加算とは２つの文字列を連結することです．たとえば

```
A$＝ "ABCD"
B$＝ "EFGH"
```

のとき，A$とB$を足して，結果をC$に入れると

```
C$＝A$＋B$
```

格納

C$には次のように格納されます．

```
?C$
ABCDEFGH
```

次のプログラムは年月日をカンマ(,)で区切って入力すると，それらをスラッシュで区切って出力します．

```
10 INPUT "ネン，ツキ，ヒ"；YY$，MM$，DD$
20 YEAR$＝YY$＋ "／"＋MM$＋ "／"＋DD$
30 PRINT YEAR$
```

● 文字列を扱う関数

文字列の一部を取り出したり，文字列の長さを返す関数もあります．まず，次のような代入をしましょう．

```
A$＝ "ABCDEFGHI"
```

9．文字列(ストリング)

文字列の長さを返す関数は LEN です．たとえば

```
?LEN (A$)
```

とすると，A$の長さ 9 が返ります．また，文字列の一部を取り出すには，次の 3 つの関数を使います．

LEFT$	文字列の左側を返す
RIGHT$	文字列の右側を返す
MID$	文字列の任意の部分を返す

たとえば

```
?LEFT$ (A$, 3)
```

を実行すると，左から 3 文字を取り出し

```
ABC
```

と表示します．また

```
?RIGHT$ (A$, 2)
```

とすると，右から 2 文字を取り出し

```
HI
```

と表示します．中間の文字列を取り出したければ MID$を使います．

```
?MID$ (A$, 3, 2)
```

とすると，文字列の 3 文字目から 2 文字を取り出し

```
CD
```

と表示します．

9．文字列(ストリング)

● 文字列の比較

キャラクタコード

文字列も数字と同じように大小を比較することができます．この比較はキャラクタコードにより行われます．たとえば〝A〞と〝B〞を比較すると，Aのキャラクタコードが41（16進）なのに対してBのキャラクタコードは42（16進）なのでBの方が大きいと判断されるのです．

次に比較の例をいくつかあげましょう．

〝AA〞 < 〝AB〞
〝MSX〞= 〝MSX〞
〝cm〞 > 〝CM〞

比較する文字列の長さが違う場合，比較が途中で終ったときは，短い方の文字列が小さくなります．

〝AB〞< 〝ABC〞
〝AB〞< 〝AB 〞

10．変数

プログラムを作るとき，変数は大変重要なものです．実際に使いながら変数について解説してみましょう．

```
PRINT "X" ↵
```

とキー入力すると

```
X
```

と表示されますが

```
PRINT X ↵
```

とキー入力すると

```
0
```

と表示されます．

これは，前者が「Xという文字を表示せよ」と命令しているのに対して，後者は「Xの中味を表示せよ」と命令しているからです．たまたまXには0が入っていたので0と表示されたのです．もし

```
X=10
```

と事前に実行していれば

```
PRINT X
10
```

となります．このXのように，その値が事前に代入文などを実行することによって変わる数を，変数と呼んでいます．

10．変数

● 変数名

変数名は次の規則にしたがって付けられています．

- 英字または数字からなり，先頭は英字でなければならない．
- 長さは何文字でも良いが，最初の２文字だけを識別する．
- 予約語が含まれてはならない．
- 小文字も大文字も同等に扱う．

予約語
P 15

正しい変数名

ub

B

BB

XYZ

X1

SUM

誤った変数名

34X（最初の文字が数字である．）

B／3（／は使えない．）

COST（予約語を含むものは使えない．）

リスト（カタカナは使えない．）

同じ名前とみなされる

XY

XYZ

XYZAB

xy

X あ Y

（最初の２文字が同じであり，大文字も小文字も同等に扱われるため．またカナや空白が変数名に含まれるものは，それらがまったく無かったものと同じになる）

10．変数

● 変数の型

変数の型は，次の２つの方法のいずれかにより決められます．

型 P 24

①型宣言文字を変数の後につける

②型宣言文

型宣言文字には次の４種類があります．

%	整数型	A%	XYZ%
!	単精度型	A!	XYZ!
#	倍精度型	A#	XYZ# A XYZ
$	文字型	A$	XYZ$

型宣言文字を省略すると，倍精度型とみなされます．

次の型宣言文（DEF 文）によっても変数の型を設定することができます．

DEFINT	整数型	DEFINT B, C
DEFSNG	単精度型	DEFSNG X-Z
DEFDBL	倍精度型	DEFDBL A, D-G
DEFSTR	文字型	DEFSTR H, I, J

DEF 文は指定した文字で始まる変数の型を設定します．上記の例では先頭１文字がA，またはDからGまでの変数が，倍精度型と定められます．

なお，型宣言文字による指定の方が，型宣言文による指定よりも優先します．

● 変数への代入

変数にデータを代入する場合，まず注意しなければならないことは，数値型変数には，数値型データしか代入できず，文字型変数には文字型データしか代入できないということです．したがって，次のような代入文は Type mismatch エラーになります．

```
A$=193
B= "12345"
```

10. 変数

ただし，型を変える関数も用意されています．

STR$（数値型データ）　数値型→文字型
VAL（文字型データ）　文字型→数値型

これらの関数を使えば，代入も可能です．

```
A$=STR$ (193)
B=VAL ("12345")
```

ただし

```
X=VAL ("ABCD")
```

としても "ABCD" が数字でないため，Xには0が代入されます．

型変換

数値型変数に型の違う数値型データを代入するときは，型の変換の問題が生じます．このとき，右辺のデータは左辺の変数の型に変換されます．たとえば

```
A%=10.5
```

とすると，小数点以下が切り捨てられ，A%には10が代入されます．次の例は，倍精度型が単精度に変換されるものです．

```
10 D!=6/7
20 PRINT D!
RUN
 .857143
```

精度

また，精度の違う数値間の演算の場合は，精度の高い方に変換されて演算が行われます．たとえば

```
A#=10#/3!
```

とすると，A#=3.3333333333333となる場合などです．

10．変数

● 配列変数

変数には1つのデータしか入れることができませんが，配列変数には複数のデータを入れることができます．

配列を使う場合は，まずDIM文によって宣言しなければなりません．たとえば

DIM A(20)

と宣言すると，A(0)～A(20)の21個の要素からなる配列がつくられます．

A	A(0)	A(1)	…………	A(19)	A(20)

配列変数にも普通の変数と同じようにいくつかの型があります．

A$(　)	文字型配列変数
A(　)	倍精度型配列変数
A#(　)	倍精度型配列変数
A!(　)	単精度型配列変数
A%(　)	整数型配列変数

配列変数の各要素にデータを入れることができます．

```
10 DIM A$(20)
20 A$(0)= "M"
30 A$(1)= "S"
40 A$(2)= "X"
50 A$(3)= " "
60 A$(4)= "B"
70 A$(5)= "A"
80 A$(6)= "S"
90 A$(7)= "I"
100 A$(8)= "C"
```

A(0)	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)	
M	S	X		B	A	S	I	C	…………

カッコ内の数字は添字と呼ばれ，変数にすることもできます．添字に変数を使うと，プログラムは簡単になります．

添字

10．変数

```
10 DIM A$(20)
20 DATA M, S, X, " ", B, A, S, I, C
30 FOR I=0 TO 8
40 READ A$(I)
50 NEXT I
```

なお配列の大きさが10以下のときは，DIM 文による宣言を省略することができます．

A(10)，B(50)など添字が１つの配列は，１次元配列と呼ばれます．一方，A(10，10)のように添字が２つある配列は，２次元配列と呼ばれます．たとえば

```
DIM A(5, 5)
```

のように宣言すると，次のように６×６の配列要素が確保されます．

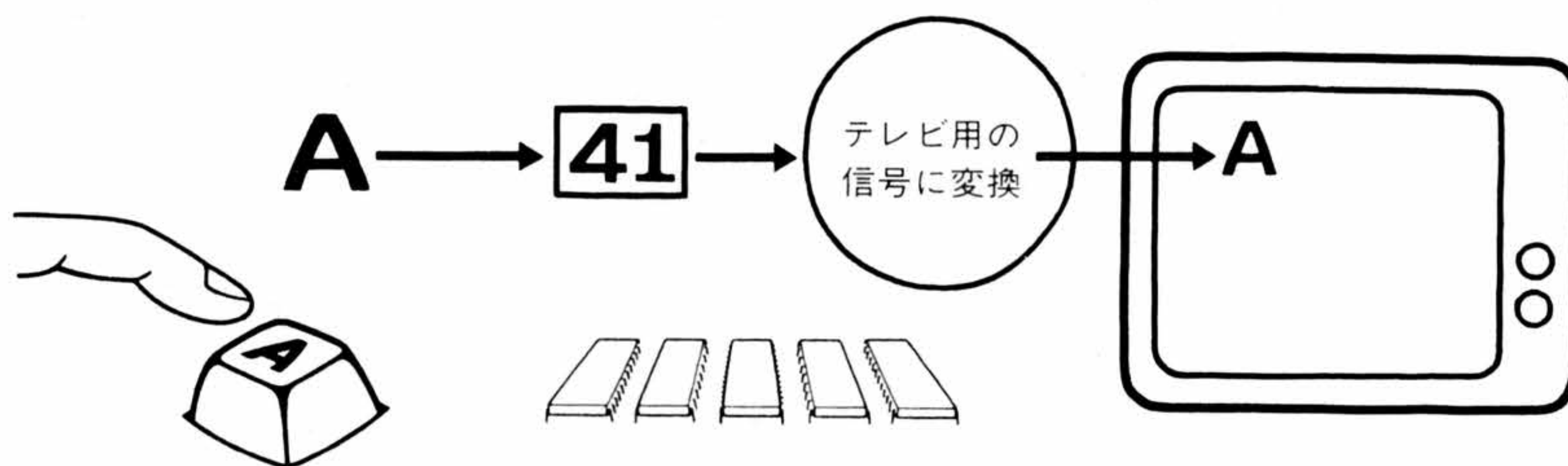
A(I, J)

I ＼ J	0	1	2	3	4	5
0						
1			A(1, 2)			
2						
3						
4						
5						

また，添字が３つものを３次元配列，n 個のものを n 次元配列と呼びます．次元は255まで可能ですが，実際はメモリの容量に制限されます．

11．キャラクタ

ディスプレイに表される文字や数字は，コンピュータの内部では，そのままの形で記憶されるのではなく，一定のコード（数字）で記憶されています．

たとえば A とタイプすると，画面にはAと表示されますが，コンピュータのメモリには，１バイトの領域に41（16進）と記憶されています．

これは，コンピュータの内部では数字しか扱えないことが理由です．Aとタイプすると，まず，キーボードのバッファにはコード41（16進数）と記憶され，画面に表示するときはコードを文字に変換しているのです．

バッファ

このように，コンピュータで扱われる文字は，内部ではすべて対応するコードによって記憶されています．そして，このコードのことをキャラクタコードと呼んでいます．

キャラクタコード

文字とキャラクタコードの対応については巻末資料を参照してください．

MSX BASICでは文字をコードに変換するための関数ASCとコードを文字に変換する関数CHR$が用意されています．

たとえば

```
?ASC("P")
```

とすると，文字Pがキャラクタコード80（10進）に変換されて

```
80
```

と表示されます．反対に

```
?CHR$(80)
```

とすると，キャラクタコード80が文字Ｐに変換されて

```
P
```

と表示されます．　次のプログラムを実行すると，コード 00 から FF(16進）に対応するすべての文字が表示されます．ただし，00〜1F（16進）までは，コントロールキャラクタなので画面には表示されません．

```
10 FOR I=0 TO 15
20 FOR J=0 TO 15
30 PRINT CHR$ (I * 16+J) ;
40 NEXT J
50 PRINT
60 NEXT I
70 END
```

● コントロールキャラクタ

コントロールキーは他の文字キーと組み合わせて，特殊な処理を行うのに使用します．たとえば，コントロールキーを押しながら G を押すと，ビープ音を発します．

このように，コントロールキーと文字キーを組み合わせたものをコントロールキャラクタと呼び，キャラクターコードの01から１Ｆ（16進）を占めます．

このコントロールキャラクタの使い方を説明しましょう．

● カーソル移動

カーソル移動キーと同等の働きをするコントロールキーは４つあります．

CTRL ＋ ¥ (1C)　→ と同じ働きをする．
CTRL ＋] (1D)　← と同じ働きをする．
CTRL ＋ ^ (1E)　↑ と同じ働きをする．
CTRL ＋ _ (1F)　↓ と同じ働きをする．（アンダーライン）

11．キャラクタ

ワード

これは，ワード単位にカーソルを動かすこともできます．

CTRL + F (06) 次のワードの先頭へカーソルを動かす．
CTRL + B (02) 前のワードの先頭へカーソルを動かす．

行の最後にカーソルを動かすには次のような操作をします．

CTRL + N (0E) 行の最後にカーソルを動かす．

● 特殊キーとして

特殊キーの代わりに使えるものもあります．

CTRL + H (08) BS
CTRL + I (09) TAB
CTRL + K (0B) HOME
CTRL + L (0C) CLS
CTRL + M (0D) RETURN
CTRL + R (12) INS
CTRL + X (18) SELECT
CTRL + [(1B) ESC

● 行の削除

行全体を削除したり，行の途中から削除することにも使うことができます．

CTRL + E (05) 行のカーソル以降を削除する．
CTRL + U (15) カーソルのある行を削除する．

● その他

その他，次のようなキャラクタコードがあります．

CTRL + C (03) INPUT 文の実行中止．
CTRL + G (07) ビープ音を鳴らす．

11．キャラクタ

グラフィック・キャラクタのヘッダーは，0〜FFのキャラクタコードでは表せないものを表示するためのコードで？CHR$（1）：CHR$（65）のように使います．これによって，31種類のキャラクタ（主に漢字）が表示できます．

```
10 FOR I=65 TO 95
20 PRINT CHR$ (1) ; CHR$(I) ;
30 NEXT
RUN
月火水木金土日年円時分秒百千万π┴┬┤├┼│─┌┐└┘╳大
中小
```

12. 音楽の演奏(PLAY)

音楽を演奏するための命令として PLAY があります. PLAY 命令ではミュージックマクロ言語を使って音を出します.

ミュージック
マクロ言語

PLAY "ミュージックマクロ言語"

主なミュージックマクロ言語には次のようなものがあります.

音程 (音符)

A～G　　オクターブ内の音程を示す.

On　　オクターブを示す.

Nn　　絶対音程を示す.

音の長さ (音符)

Ln　　音の長さを示す.

休符

Rn　　休みの長さを示す.

テンポ

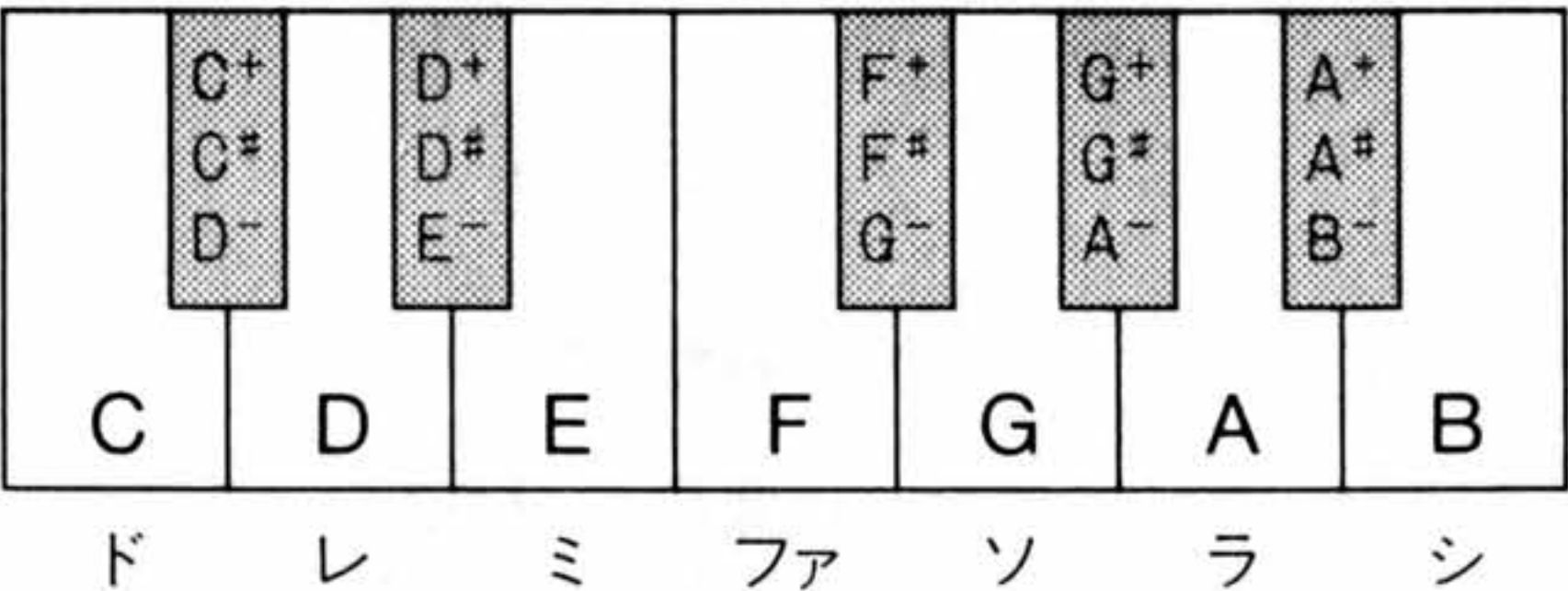
Tn　　演奏のテンポを示す.

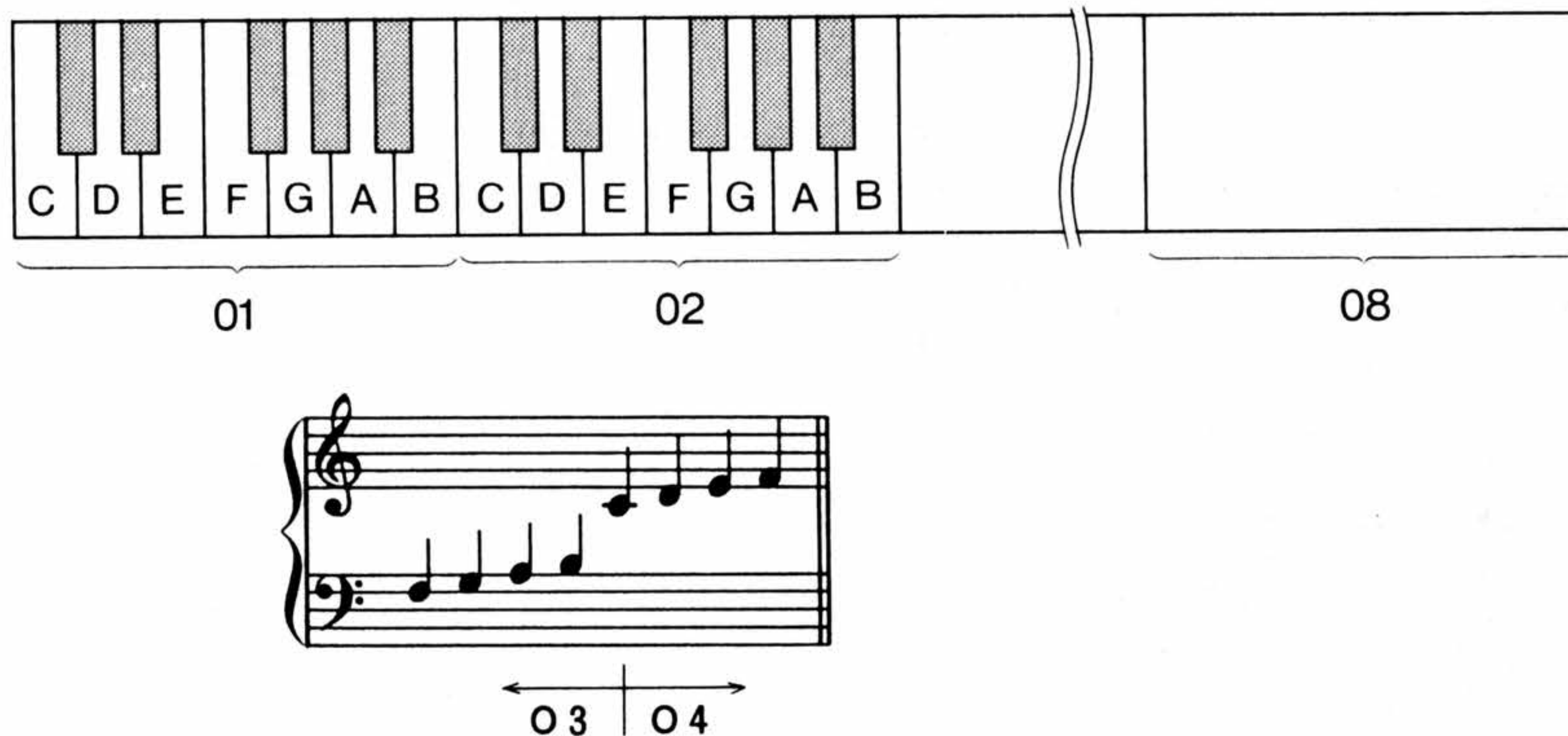
● 音程

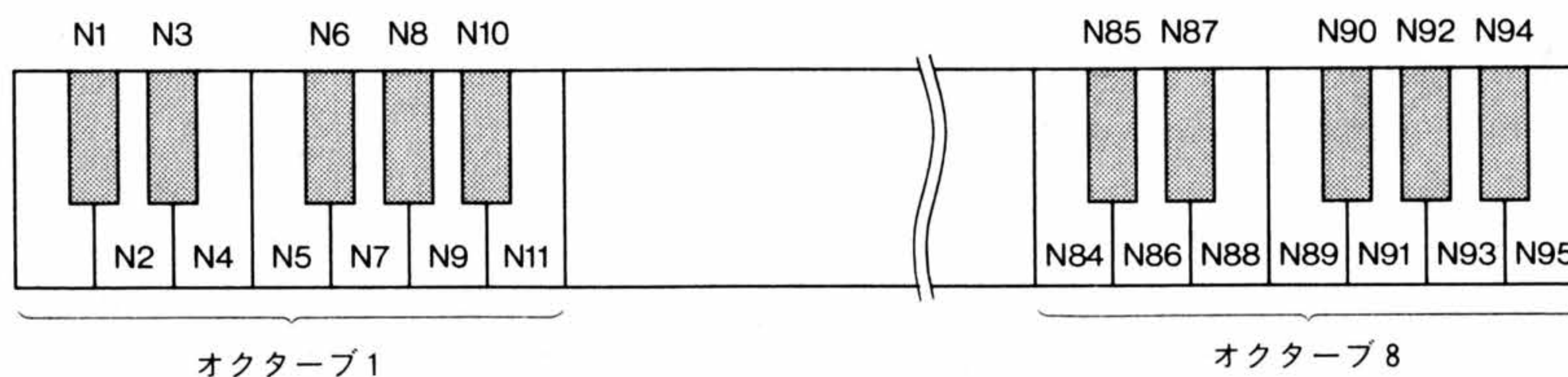
ド～シの音を出すのにそれぞれ cdefgab を指定します.

PLAY "cdefgab"

とすれば, ドレミファソラシと演奏します. アルファベットは大文字でも小文字でもかまいません.

また, 半音上げるには#または＋を, 半音下げるには－をアルファベットの後につけます.

12．音楽の演奏(PLAY)

オクターブはOnによって1から8まで指定することができます．省略値は4です．

A～Gはオクターブ内の相対音程を示していますがNnを使って，絶対音程を示すこともできます．

Nnと音程との対応は次の通りです．オクターブ4のドはN36です．

● 音の長さ

音の長さはLnによって指定します．nは1から64で，nの値と音の長さとの対応は次の通りです．

𝅝（全 音 符）＝L1　　𝅗𝅥（2分音符）＝L2

♩（4分音符）＝L4　　♪（8分音符）＝L8

𝅘𝅥𝅯（16分音符）＝L16　　𝅘𝅥𝅰（32分音符）＝L32

𝅘𝅥𝅱（64分音符）＝L64

12．音楽の演奏(PLAY)

これらの1.5倍の長さを表わすには音程の後にピリオド(.)をつけます．たとえば符点4分音符は次のようにして示します．

(符点4分音符)＝L 4 音程.

2つ以上のピリオドをつけることもできます．たとえば3つピリオドを付けると，その音の長さは，元の長さの1.5×1.5×1.5＝3.375倍になります．

● 音符

音符は音程と音の長さによって表すことができます．

PLAY "L 4 O 4 G." または PLAY "L 4 N 4 3."

PLAY "L 8 O 4 D+" または PLAY "L 8 N 3 9"

このように音符を示す場合，Lを最初に指定しなければなりません．ただし，Lを省略して長さnを音程の後に指定することもできます．

PLAY "L 4 O 4 G."＝PLAY "O 4 G 4."

音符1つに対応して，A〜G（またはNn）は必ず指定しなければなりません．しかし，長さやオクターブは省略することもできます．そのときは，最近にLやOで指定したものが省略値になります．

● 休符

休符はRnによって示します．nは休符の長さを示しています．

(全 休 符)＝R1	(2分休符)＝R2
(4分休符)＝R4	(8分休符)＝R8
(16分休符)＝R16	(32分休符)＝R32
(64分休符)＝R64	

音の長さと同じように，ピリオドを使って1.5倍長い休符を示すことができます．たとえば符点4分休符は次のように示します．

(符点4分休符)＝R 4.

12. 音楽の演奏(PLAY)

● テンポ

演奏のテンポは Tn で示します．n は 1 分間に演奏される 4 分音符の数を示します．n は32から255の値をとり，省略値は120です．

● 演奏

ここで，峠の我が家を演奏してみましょう．

```
10 PLAY "T128"
20 PLAY "O4L4CCFGL2AL8FED4. B-B-4"
30 PLAY "B-2AB-O5C2O4FFF4. EF4G2."
40 PLAY "G2L4CCFGA2L8FED4. B-8B-4"
50 PLAY "B-2L8B-B-A4. GF4E4. FG4F2."
60 END
```

12．音楽の演奏(PLAY)

● 和音

PLAY コマンドを使って，和音を出すこともできます．たとえば

の音を出したければ

```
PLAY "C", "E", "G"
```

とすればいいのです．

● 変数

ミュージックマクロ言語の中で変数を使うこともできます．このときは変数を＝と；で囲んでください．

＝〈変数〉；

たとえば

```
PLAY "N10"
```

とする代わりに

```
X=10
PLAY "N=X;"
```

とすることができます．

次のプログラムを実行すると，すべての音程の音を出すことができます．

```
10 FOR I= 1 TO 96
20 PLAY "N=I;"
30 NEXT I
```

12．音楽の演奏(PLAY)

● 音量

その他，次の3つのミュージックマクロ言語があります．

Vn	音の大きさを指定する．
Sn	エンベロープの形を変える．
Mn	エンベロープの周期を変える．

SとMはペアで使われます．また，SとMはVと同時に使うことはできません．

次のプログラムを実行すると，ボリュームを変えて，ドの音を連続して出すことができます．

```
10 FOR V=0  TO 15
20 PLAY  "V=V; C"
30 NEXT  V
```

エンベロープ
P 51

SとMによってエンベロープを指定すると，音量が周期的に変化します．そのときの音量の揺れる形を指定するのがSで，振れる速さを指定するのがMです．

次のプログラムは，音の形や揺れの速さを変えて，ドの音を出します．

```
10 FOR S= 1  TO 15
20 FOR M=500 TO 1500 STEP 100
30 PLAY  "S=S; M=M; C"
40 NEXT M
50 NEXT S
```

エンベロープについては「SOUND機能」の項を参照してください．

13．SOUND機能

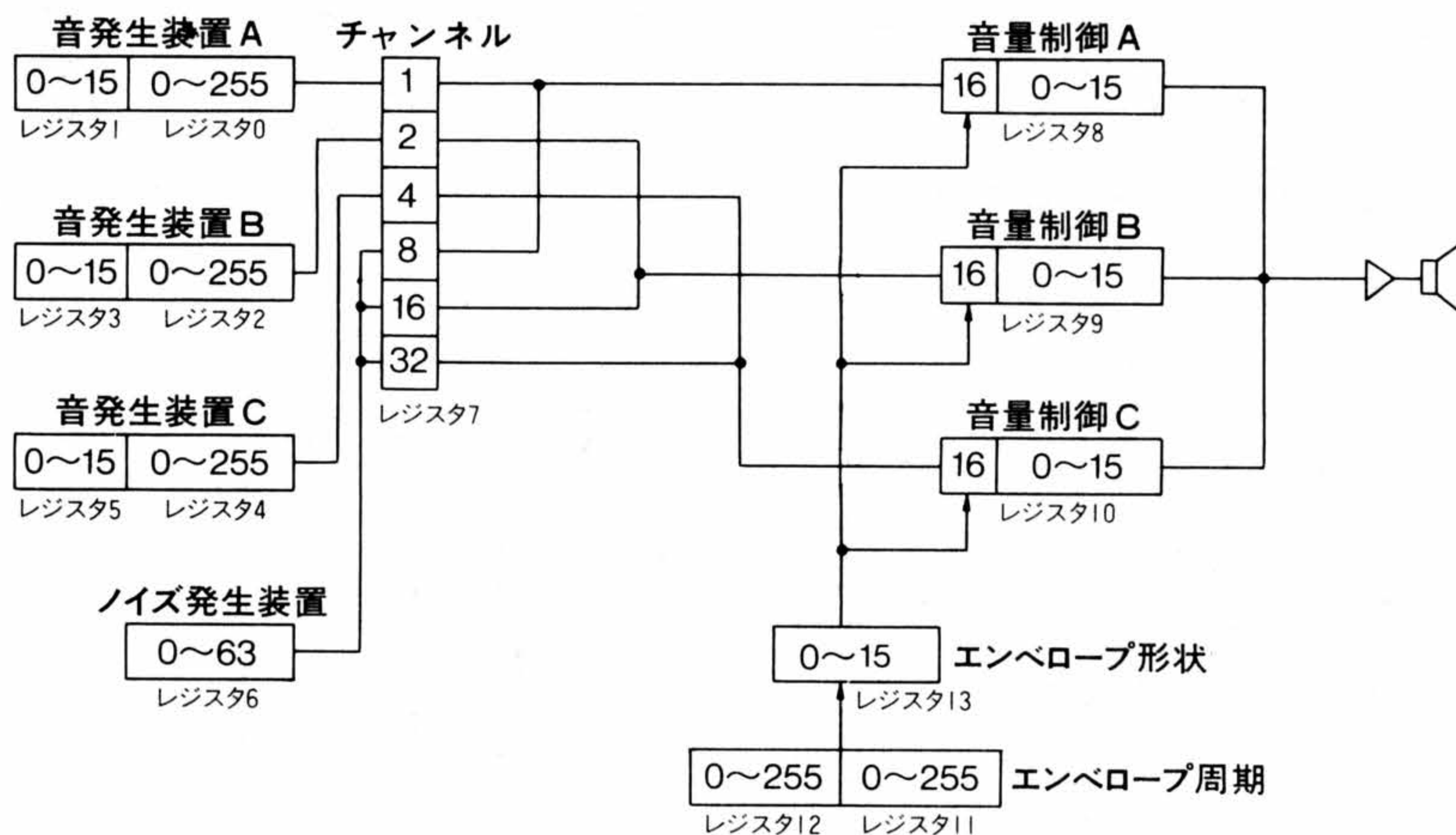
SOUND命令は，SOUND IC（PSG）にある16個のレジスタにデータを入れることにより，音を発生させる命令です．

PSG
レジスタ
P 102

SOUND　レジスタ番号，値

SOUND IC内のレジスタの構成は次のようになっています．

レジスタ ＼ ビット		B7	B6	B5	B4	B3	B2	B1	B0
R 0	チャンネルA周波数	8BIT FT A							
R 1						4BIT CT A			
R 2	チャンネルB周波数	8BIT FT B							
R 3						4BIT CT B			
R 4	チャンネルC周波数	8BIT FT C							
R 5						4BIT CT C			
R 6	ノイズ周波数				5BIT NP				
R 7	チャンネル設定	$\overline{IN}$/OUT		$\overline{NOISE}$			$\overline{TONE}$		
		IOB	IOA	C	B	A	C	B	A
R 8	チャンネルA音量				M	L3	L2	L1	L0
R 9	チャンネルB音量				M	L3	L2	L1	L0
R10	チャンネルC音量				M	L3	L2	L1	L0
R11	エンベロープ周期	8BIT FT							
R12		8BIT CT							
R13	エンベロープ周期					E3	E2	E1	E0
R14	I/OポートAデータ	8BIT パラレル I/OポートA							
R15	I/OポートBデータ	8BIT パラレル I/OポートB							

13. SOUND機能

● 音の発生

音の発生装置と音量の制御装置は3チャンネルずつに分離されているので，まったく独立に制御することができます．

チャンネルは該当するビットが0のとき入力を受け付け，1のとき無視します．音量は0にすると無音，15にすると最大の音量となります．

音の周波数設定レジスタR0, R1（チャンネルAの場合）に入れる値FT，CTは次式によって計算されます．

$$TP=\frac{fclock}{16\times ft} \qquad CT+\frac{FT}{256}=\frac{TP}{256}$$

ここで

fclock　1.78977 MHz

ft　出力する周波数

FT　FTA（R0）の値（10進で0〜255）

CT　CTA（R1）の値（10進で0〜15）

13. SOUND機能

次のプログラムを実行すると，チャンネルAを通してすべての音を出すことができます．

```
10 SOUND 7, 56
20 FOR I=0 TO 255
30 FOR J=0 TO 15
40 SOUND 0, I : SOUND 1, J
50 FOR K=1 TO 15
60 SOUND 8, K
70 NEXT K
80 NEXT J
90 NEXT I
```

40行と60行を次のように変え，その他のレジスタにデータを入れると，それぞれ，チャンネルB，チャンネル Cから音を発します．

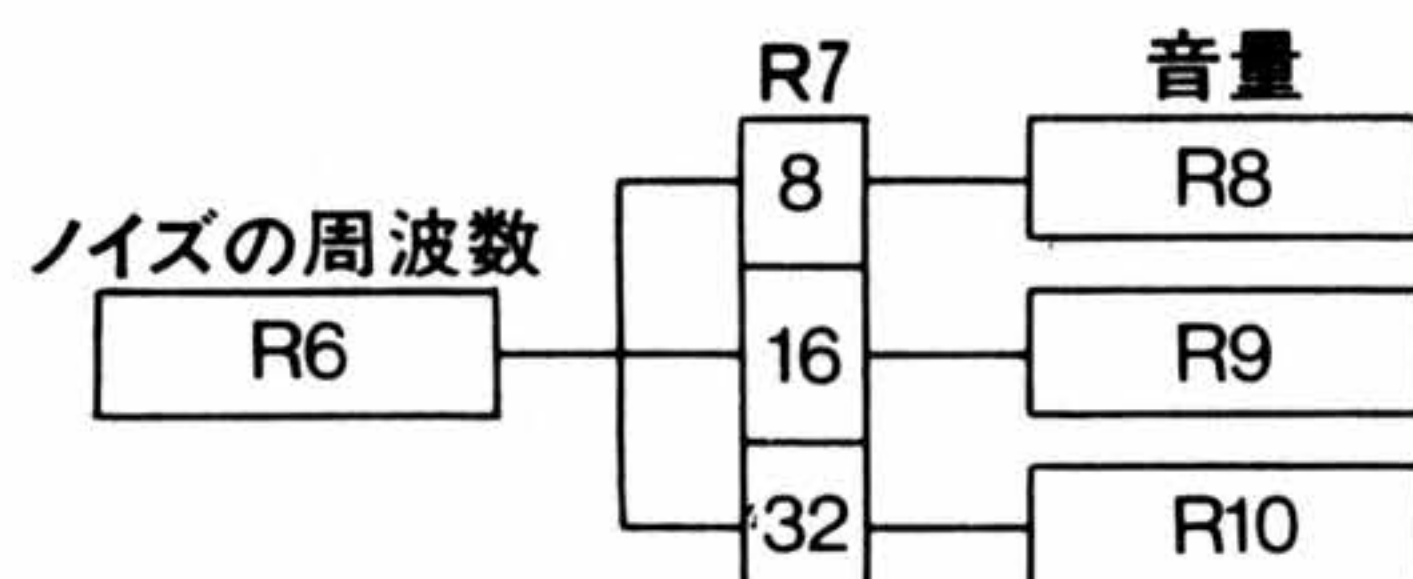
```
40 SOUND 2, I : SOUND 3, J  ⎫
60 SOUND 9, K               ⎭ チャンネルB
40 SOUND 4, I : SOUND 5, J  ⎫
60 SOUND 10, K              ⎭ チャンネルC
```

次のプログラムを実行すると，チャンネルA，B，Cから同時に音を発します．

```
10 SOUND 0, 0 : SOUND 1, 1
20 SOUND 2, 128 : SOUND 3, 1
30 SOUND 4, 128 : SOUND 5, 0
40 SOUND 8, 6 : SOUND 9, 9 : SOUND 10, 12
50 SOUND 7, 56
```

13. SOUND機能

● ノイズの発生

ノイズの発生装置は1つしかないので，各チャンネルで共通に使うことになります．

ノイズ周波数設定レジスタR6に入れる値NP_{10}は次式により計算できます．

$$NP = \frac{fclock}{16 \times fN}$$

ここで

fclock　1.78977 MHz

fN　　出力したいノイズ周波数（Hz）

NP　　NP（R6）に設定する値（10進で0～31）

次のプログラムもチャンネルAを通して，すべてのノイズを発するものです．

```
10 SOUND 7, 7
20 FOR I=0 TO 31
30 SOUND 6, I
40 FOR J=1 TO 15
50 SOUND 8, J
60 NEXT J
70 NEXT I
```

50行を次のように変えると，それぞれ，チャンネルB，チャンネルCからノイズを発生します．聞こえる音はまったく同じです．

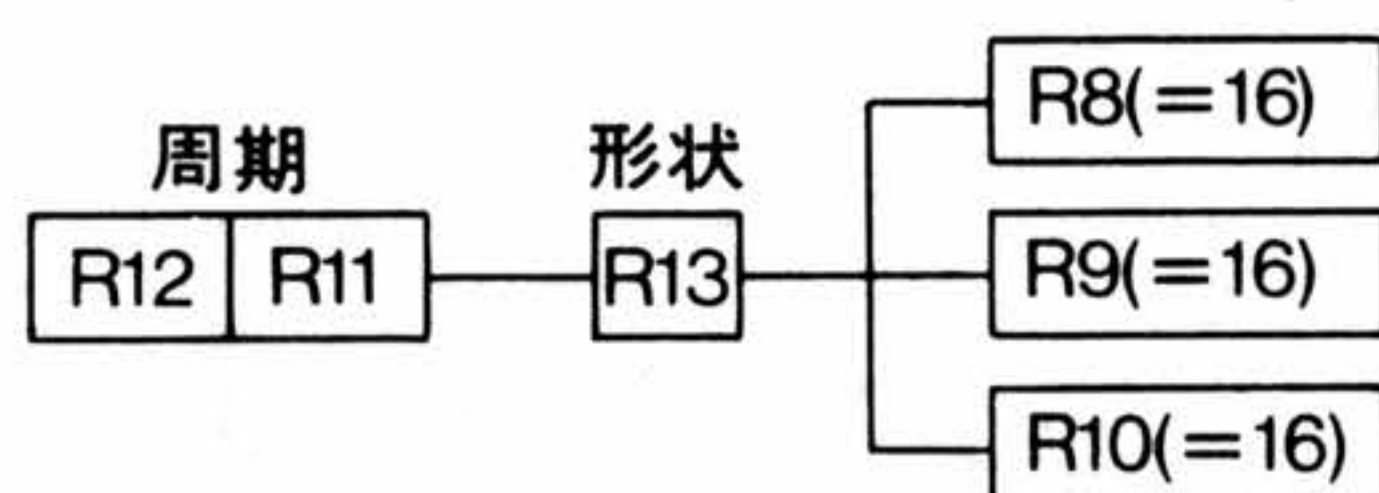
50 SOUND 9, J……チャンネルBから発生

50 SOUND 10, J……チャンネルCから発生

ノイズと音は混ぜて発することもできます．

13．SOUND機能

● エンベロープの発生

レジスタ8，9または10の値を16にすることによって，音量が時間的に変化するエンベロープモードを設定できます．このときレジスタ13にエンベロープの形状，レジスタ11，12にエンベロープの周期をセットします．

エンベロープの周期は次式で計算できます．

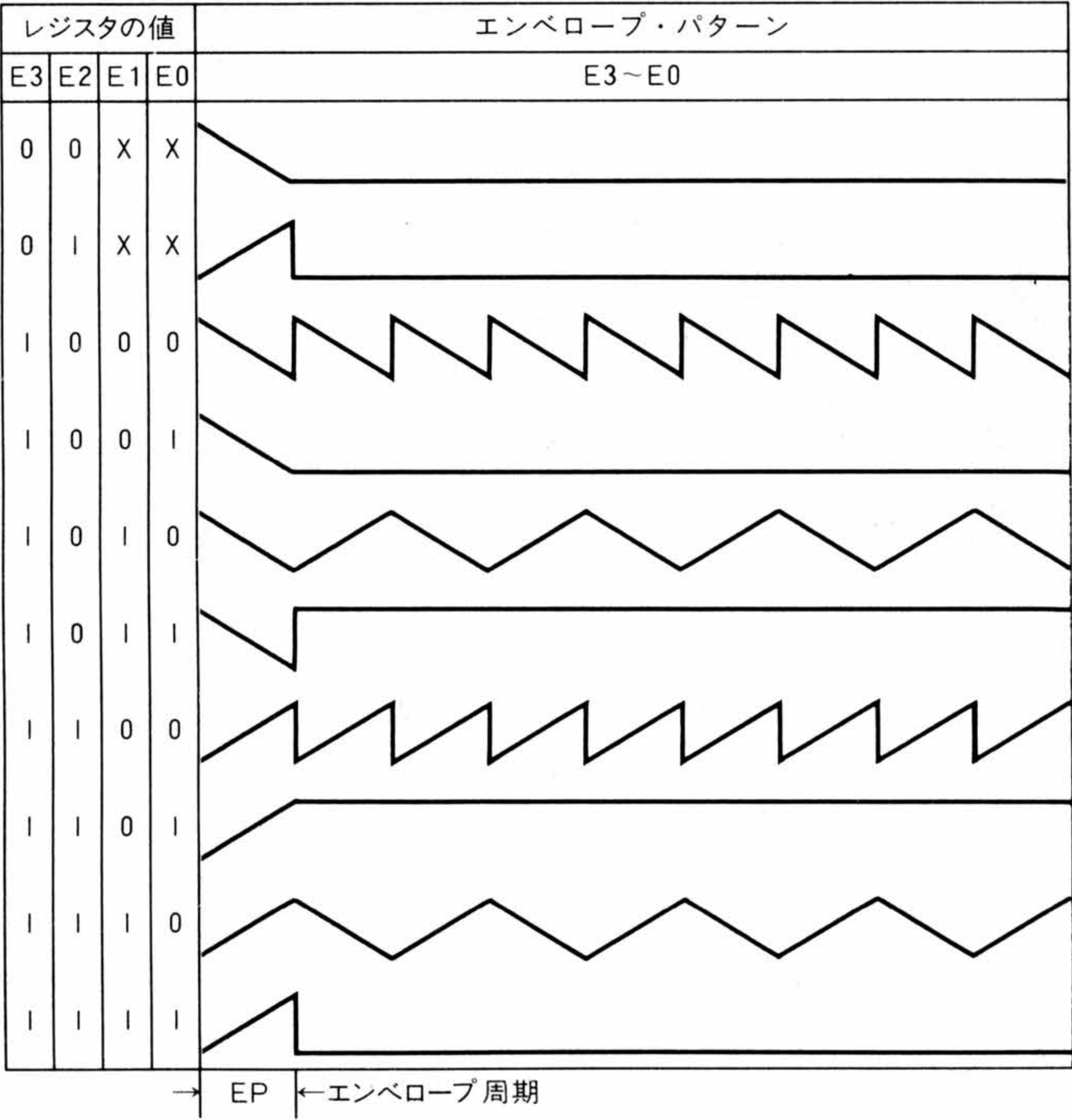
$$EP=\frac{fclock}{256fE} \qquad CT+\frac{FT}{256}=\frac{EP}{256}$$

ここで

fclock	1.78977 MHz
EP	音が増加または減衰する周波数 Hz
FT	FT（R11）に設定する値（10進で0～255）
CT	CT（R12）　〃

エンベロープの形状は，レジスタ13のE0からE3の値で設定します．E0からE3の値がどのような波形に対応しているかを次頁の図に示します．なお，図中のXは0でも1でもかまわないことを意味します．

13. SOUND機能

エンベロープ発生器出力

次のプログラムはレジスタ8を通して，ヘリコプターのような音を発生させるものです.

```
10 FOR I=0 TO 13
20 READ A:SOUND I,A
30 NEXT
40 DATA 20,0,30,0,0,9,0
50 DATA 48,16,4,6,100,2,12
```

14. 画面操作

画面モードには大きく分けて，テキストモードとグラフィックモードがあります．テキストモードは文字を画面に表示させるときのモードで，グラフィックモードは点や線や図形を画面に表示させるモードです．

また，テキストモードには，40桁×24桁のモードと32桁×24行のモードがあります．グラフィックモードには，高分解能グラフィックモードと，低分解能グラフィックモードがあります．

テキスト
グラフィック
分解能

テキストモード

40桁×24行
32桁×24行

グラフィックモード

高分解能
低分解能

これらのモードはSCREEN 命令で指定することができます．

```
SCREEN 〈モード〉
```

モード　0：40×24テキストモード
　　　　1：32×24テキストモード
　　　　2：高分解能グラフィックモード
　　　　3：低分解能グラフィックモード

テキストモードでは，PUT SPRITE を除くすべてのグラフィック命令は使うことができません．また，グラフィックモードはプログラムモード（プログラム実行中）の場合のみ有効です．プログラムが終了すると，自動的にテキストモードに戻ります．プログラム中のINPUT 命令によってもテキストモードに戻ります．

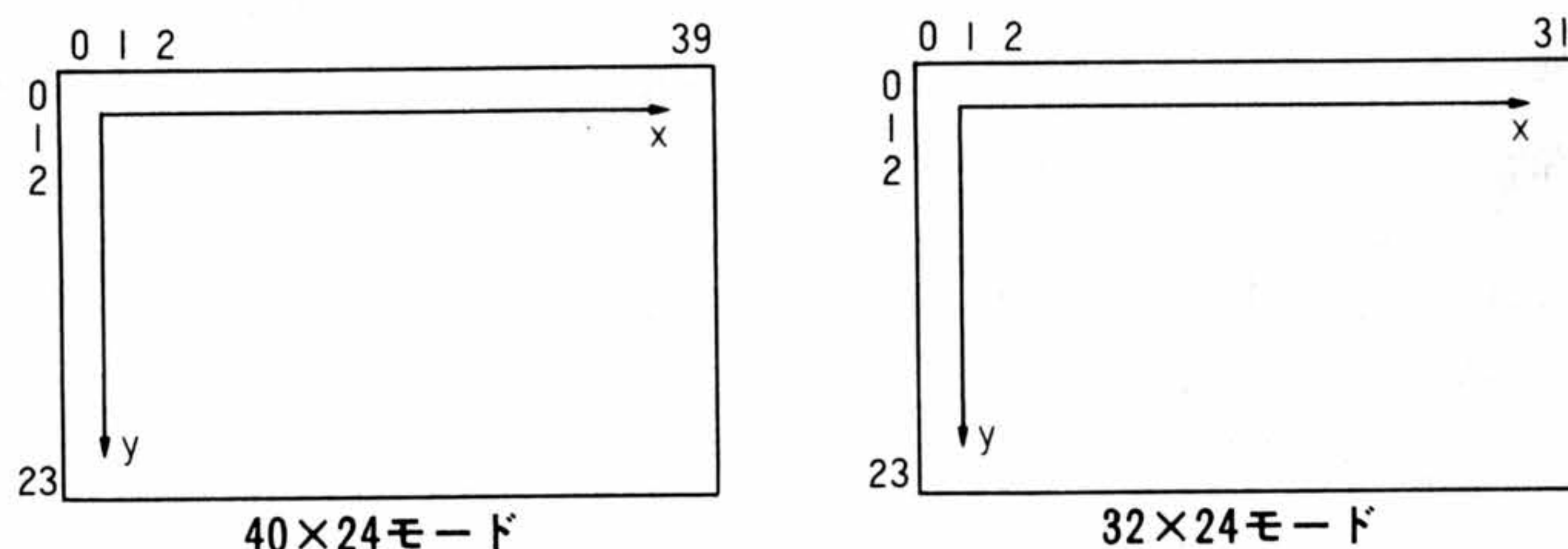
14. 画面操作

1 テキストモード

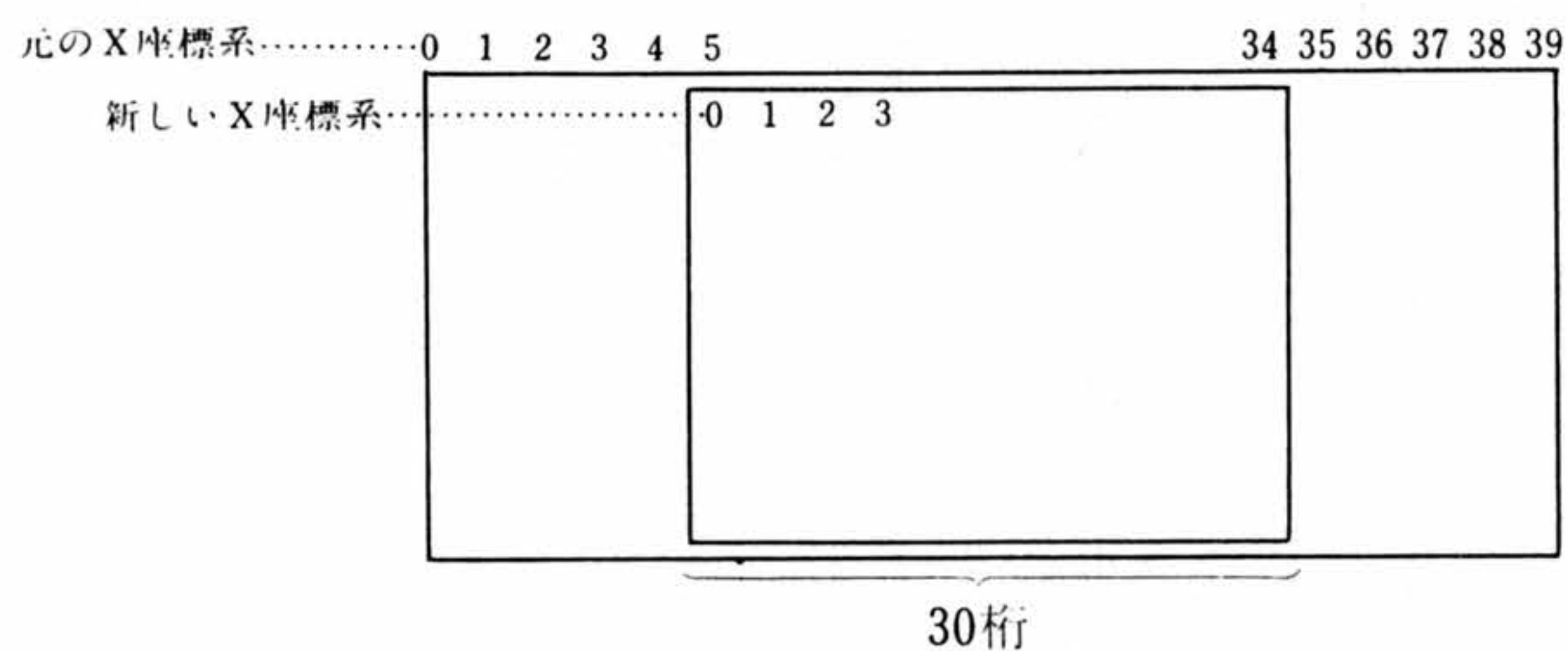
すでに述べたように，テキストモードには40×24のモードと32×24のモードがあります．それぞれの座標系は次のように設定されています．

座標系

40×24モード

32×24モード

WIDTH 文によって，画面の幅を制限することもできます．たとえば40×24モードで

```
WIDTH 30
```

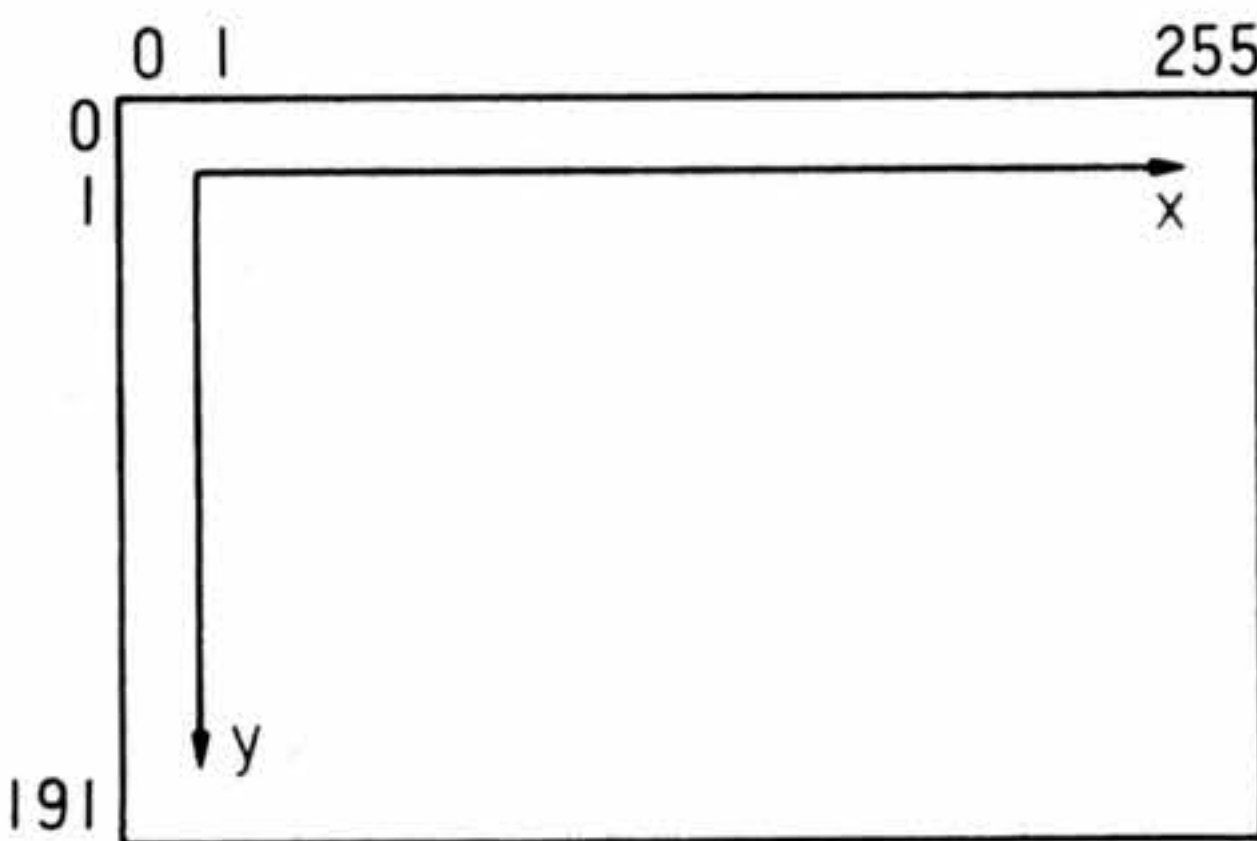
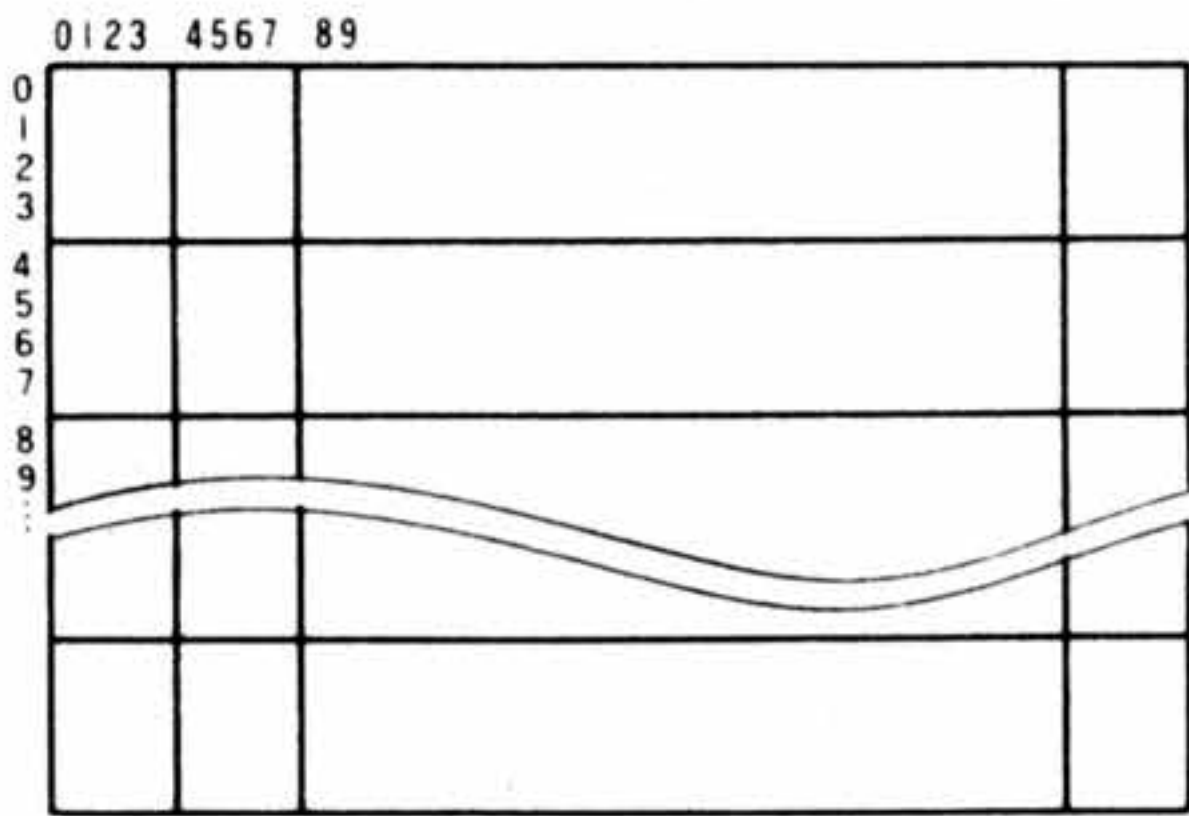
を実行すると，次のように座標系が変わります．

テキストモードでの画面表示関係の命令は次の通りです．

PRINT	画面に文字を表示する．
LOCATE	カーソルを位置づける．
CLS	画面をクリアする．
COLOR	文字の色などを指定する（詳しくはグラフィックモードを参照）．

14. 画面操作

2　グラフィックモード

グラフィック画面は，横256ドット，縦192ドットの分解能を持ち，次のように座標が設定されています．

ドット

高分解能モードでは，各ドットが1つの点に対応しますが，低分解能では4×4ドットが1つの点となります．したがって，低分解能の解像度は64×48となります．

15．絵の描き方

スクリーンモード２や３では，絵を描くためのいろいろなコマンドが使えます．絵を描くということは，色を決めて点や線，円を描くことなどです．それぞれについて解説していきます．

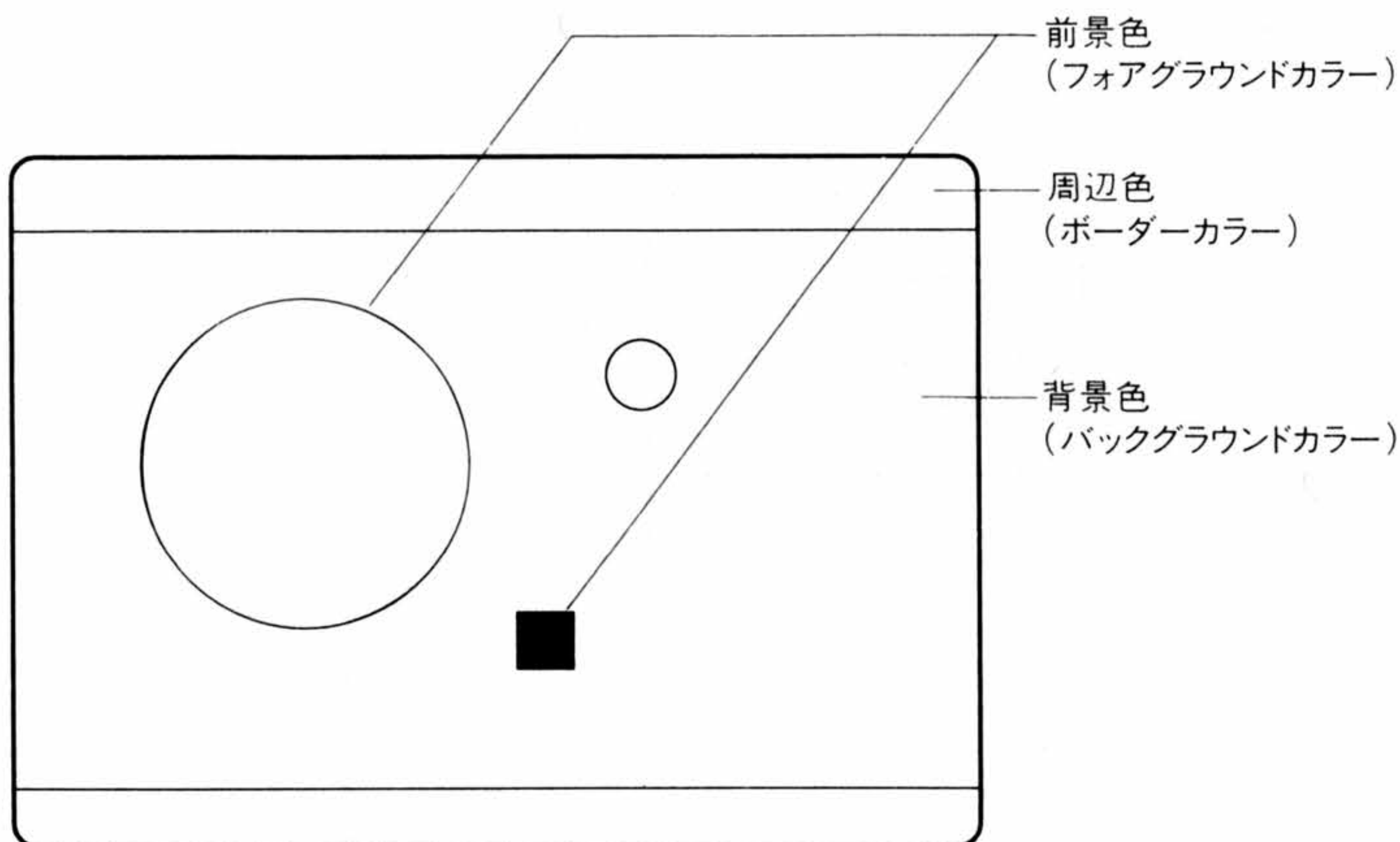
● 色

色は16色使うことができます．カラーコードと色の対応は次の通りです．

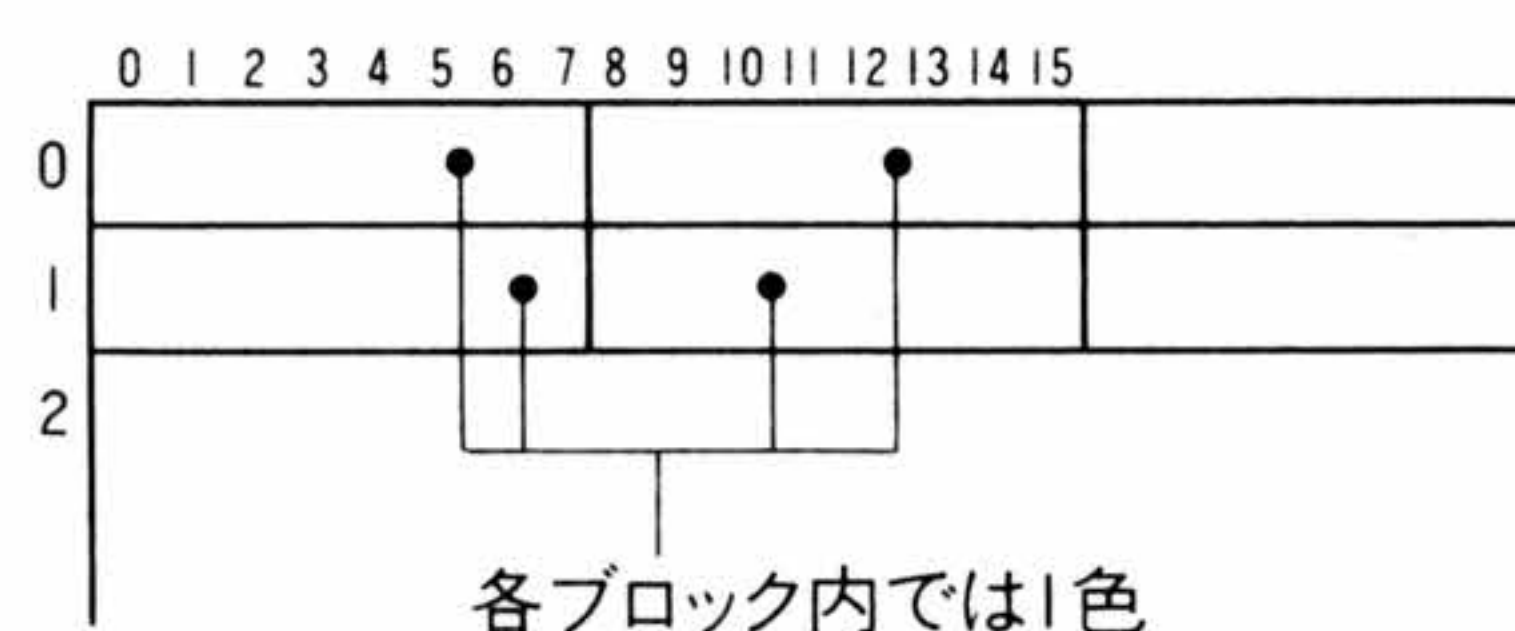
0：透明	4：暗い青	8：赤	12：暗い緑
1：黒	5：明るい青	9：明るい赤	13：紫
2：緑	6：暗い赤	10：黄	14：灰
3：明るい緑	7：水色	11：明るい黄	15：白

このカラーコードを指定して，画面の基本的な色を設定するのがCOLOR文です．

前景色
周辺色
背景色

前景色
（フォアグラウンドカラー）
周辺色
（ボーダーカラー）
背景色
（バックグラウンドカラー）

前景色はグラフィック命令を使っても指定することができます．ただし，高分解能モード（SCREEN2）では横８ドットのブロック内では，前景色として１色しか指定できないので気を付けてください．

15．絵の描き方

たとえば，座標（3，0）に青い点を打ち，次に座標（6，0）に赤い点を打つと，座標（3，0）の色も赤に変わってしまいます．このように同じブロック内では最後に指定した色が有効になるのです．

● 点を打つ

点を打つにはPSET命令を使います．たとえば座標（5，10）に点を打つには

```
PSET (5, 10)
```

とすればよいのです．点に色をつけることもできます．赤い色（8）をつけたければ

```
PSET (5, 10), 8
```

とします．

次のプログラムは画面にランダムに点を打つものです．色も変わります．

```
10 SCREEN 2
20 X=RND (1) * 255 : Y=RND (1) * 191 : C=INT (RND (1)
* 16)
30 PSET (X, Y), C
40 GO TO 20
```

10行を次のように変えると

```
10 SCREEN 3
```

低分解能モードで点を打ちます．このとき，4×4ドットの同じブロック内なら，どの座標を指定しても同じ位置に点が打たれます．たとえば次の2つ

```
PSET (0, 1), 15
PSET (3, 2), 15
```

は，同じ左上角に4×4ドットの点を打つことになります．

15．絵の描き方

打たれた点を消すのは PRESET 命令です．たとえば，（5，10）に打たれた点を消すには

```
PRESET (5, 10)
```

とします．

カラーコードを省略すると，PSET 文では前景色を指定したとみなされ，PRESET 文では背景色を指定したとみなされます．つまり COLOR 15，4，7 のときなら，PSET (X，Y) は PSET (X，Y)，15 と同じ，PRESET (X，Y) は PRESET (X，Y)，4 と同じになります．

● 線や四角形を描く

線を引く場合は LINE 文を使います．たとえば座標（10，10）から（20，20）に線を引くときは

```
LINE (10, 10)- (20, 20)
```

とします．赤い色をつけたければ，カラーコード 8 を指定します．

```
LINE (10, 10)- (20, 20), 8
```

指定した 2 点を対角線とする四角形を描くこともできます．そのときは，カラーコードの後に B と指定します．

```
LINE (10, 10)- (20, 20), 8, B
```

四角形の中を塗りつぶしたければ，B の代わりに BF を使います．

```
LINE (10, 10)- (20, 20), 8, BF
```

次のプログラムは格子状に色を変えて線を描きます．

```
10 SCREEN 2
20 FOR X=0 TO 255 STEP 10
30 C=C+1
40 IF C=>15 THEN C=1
50 LINE (X, 1)- (X, 191), C
60 NEXT X
```

15. 絵の描き方

```
70 FOR Y=1 TO 191 STEP 10
80 C=C+1
90 IF C>15 THEN C=1
100 LINE (0, Y)- (255, Y), C
110 NEXT Y
120 GOTO 20
```

● 円を描く

円を描くときはCIRCLE文で円の中心の座標と半径を指定します．たとえば中心（60，50），半径10の円を描きたければ

```
CIRCLE (60, 50), 10
```

とします．カラーコードを指定して色を付けることもできます．

```
CIRCLE (60, 50), 10, 8
```

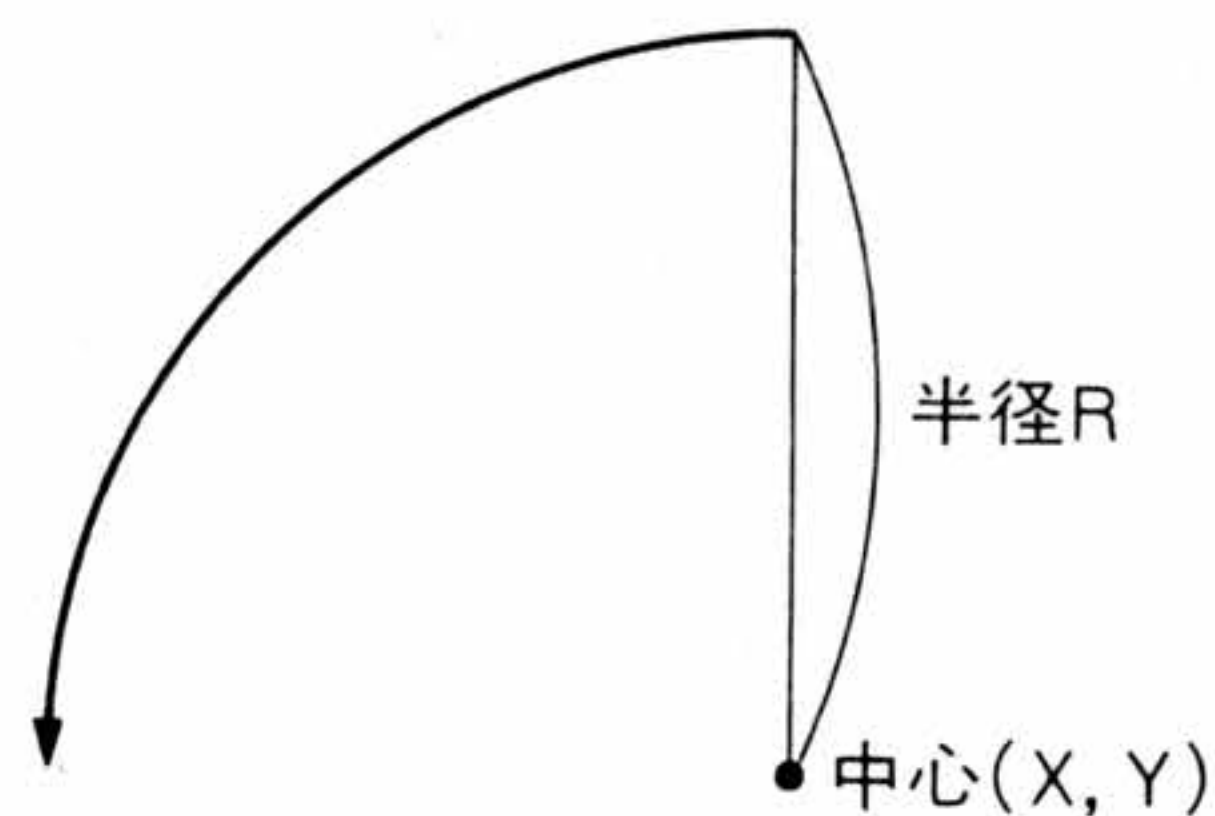
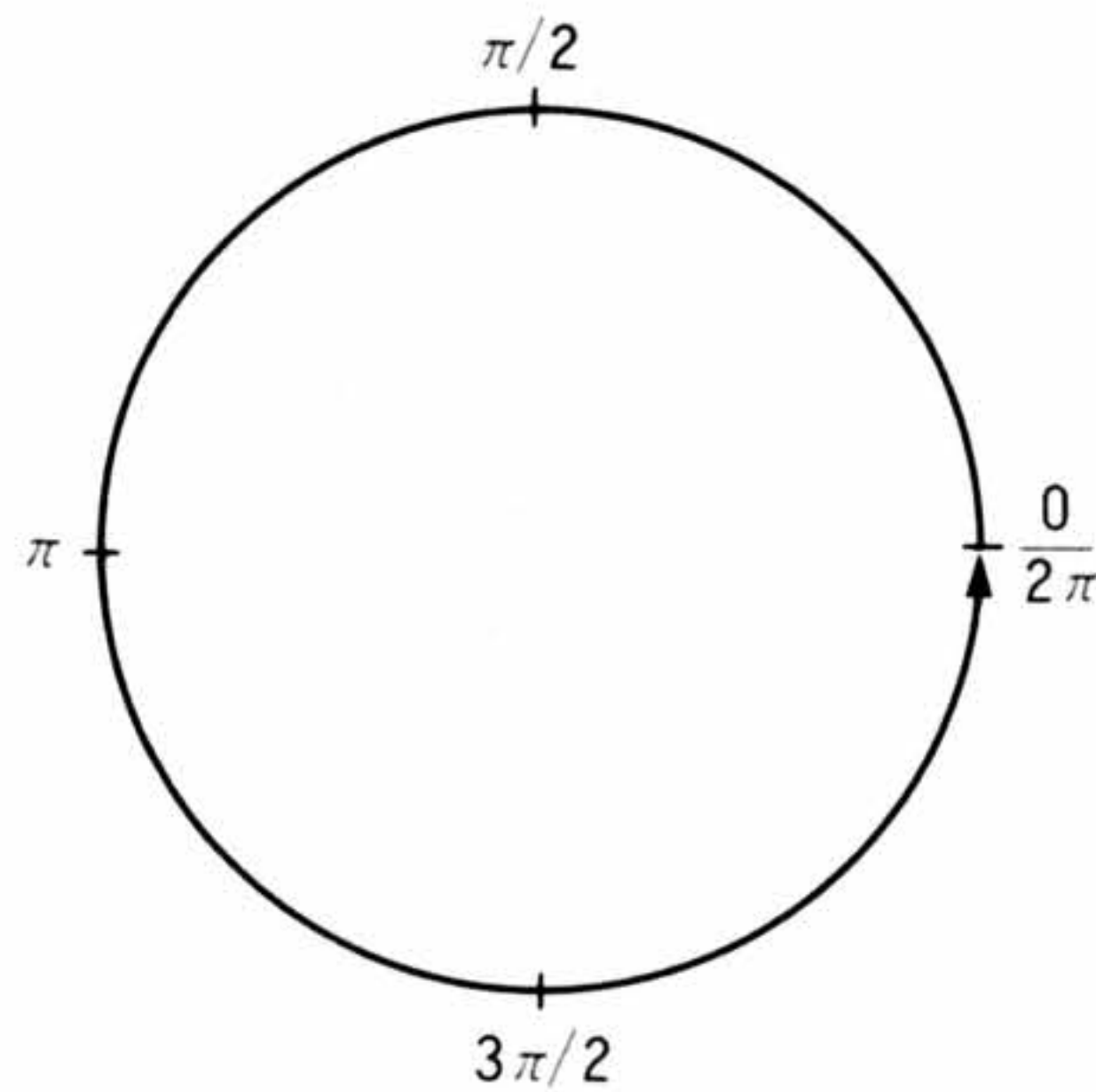
とすれば，赤い色の円が描かれます．

次のプログラムは乱数を使って，色々な円を描くものです．

```
10 SCREEN 2
20 X=RND (1) * 255 : Y=RND (1) * 151 : R=RND (1) * 50 :
 C=(RND (1) * 15) +1
30 CIRCLE (X, Y), R, C
40 GOTO 20
```

CIRCLE命令を使うと円弧を描くこともできます．そのためには円弧の開始角度と終了角度を指定します．たとえば右のような円弧を描きたいとします．

プログラムは次のようになります

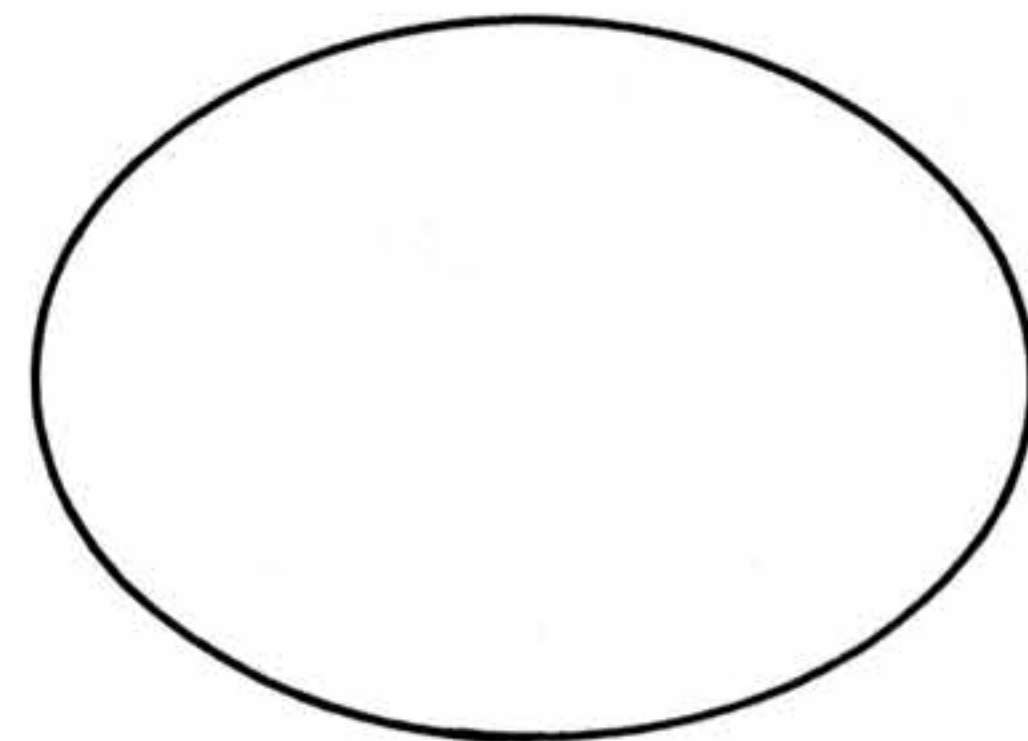
```
10 SCREEN 2
20 X=128 : Y=96 : R=80 : PI=3.14
30 CIRCLE (X, Y), R,, PI/2, PI
40 GOTO 40
```

15．絵の描き方

角度の指定はラジアンなので，360°は2π（≒6.2831853071796）になります．

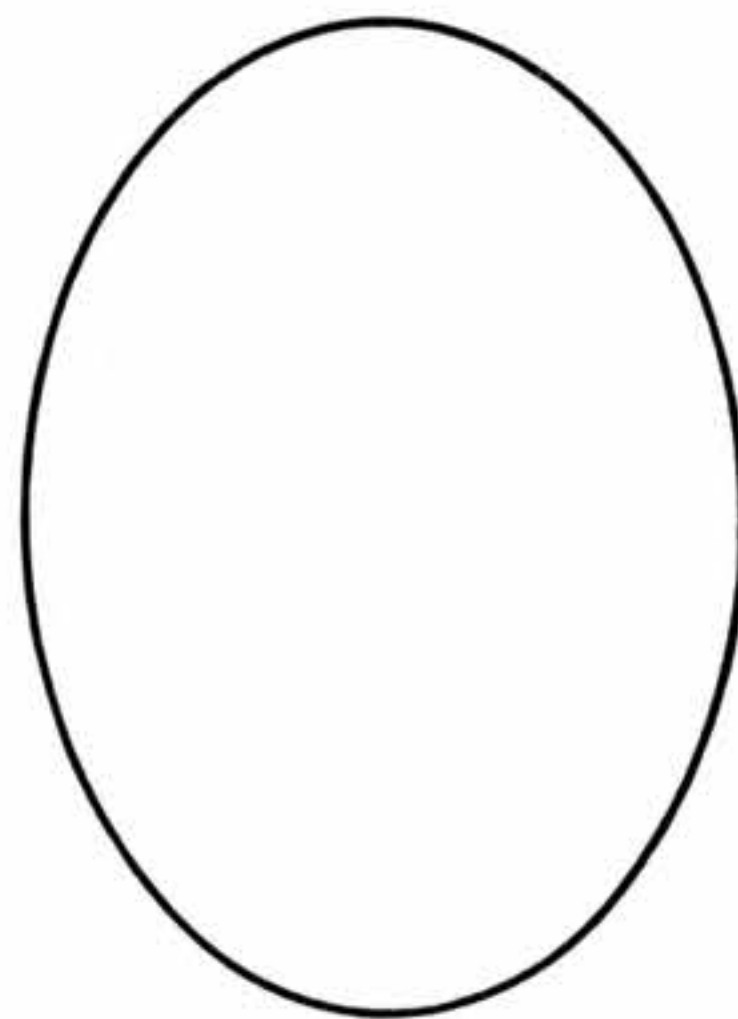
つまり開始角度が0，終了角度が2πのときは円を描くことができます．CIRCLE文でこれらのパラメータを省略したときは開始角度＝0，終了角度＝2πになっていると考えればよいでしょう．

CIRCLE文にはもう1つパラメータがあります．これを使って楕円を描くことができます．省略値は1ですが，テレビモニタの性質や1ドットのたて横比の関係でこのままでは正円にはなりません．少し横長になります．

CIRCLE（128，96），80,,,，1
（CIRCLE（128，96），80と同じ）

他の値を使ってみましょう．

CIRCLE（128，96），80,,,，2

たてのドット数(半径80)／横のドット数が2になります．

CIRCLE(128，96)，80,,,，0.5なら横長の円となります．CIRCLE(128，96)，80,,,，1.18位で正円に見えると思いますが，モニタの性質も違うかも知れませんのでいろいろと試してみてください．

15. 絵の描き方

● 色を塗る

境界線の中に色を塗るにはPAINT文を使います．境界線の中の任意の一点の座標と，境界の色，塗る色を指定します．ただし高分解能モードでは，境界の色と塗る色は同じでなければなりません．

次のプログラムは，円を描き，その中を同じ色で塗りつぶすものです．

```
10 SCREEN 2
20 CIRCLE (128, 100), 50, 8
30 PAINT (100, 100), 8
40 GOTO 40
```

低分解能モードでは，境界の色と塗る色を変えることもできます．

```
10 SCREEN 3
20 CIRCLE (128, 100), 50, 8
30 PAINT (100, 100), 5, 8
40 GOTO 40
```

● 任意の図形を描く

DRAW命令を使うと，四角でも三角でも自由に描くことができます．

DRAW命令では次のようなグラフィックマクロ言語を使って線を描きます．

グラフィックマクロ言語

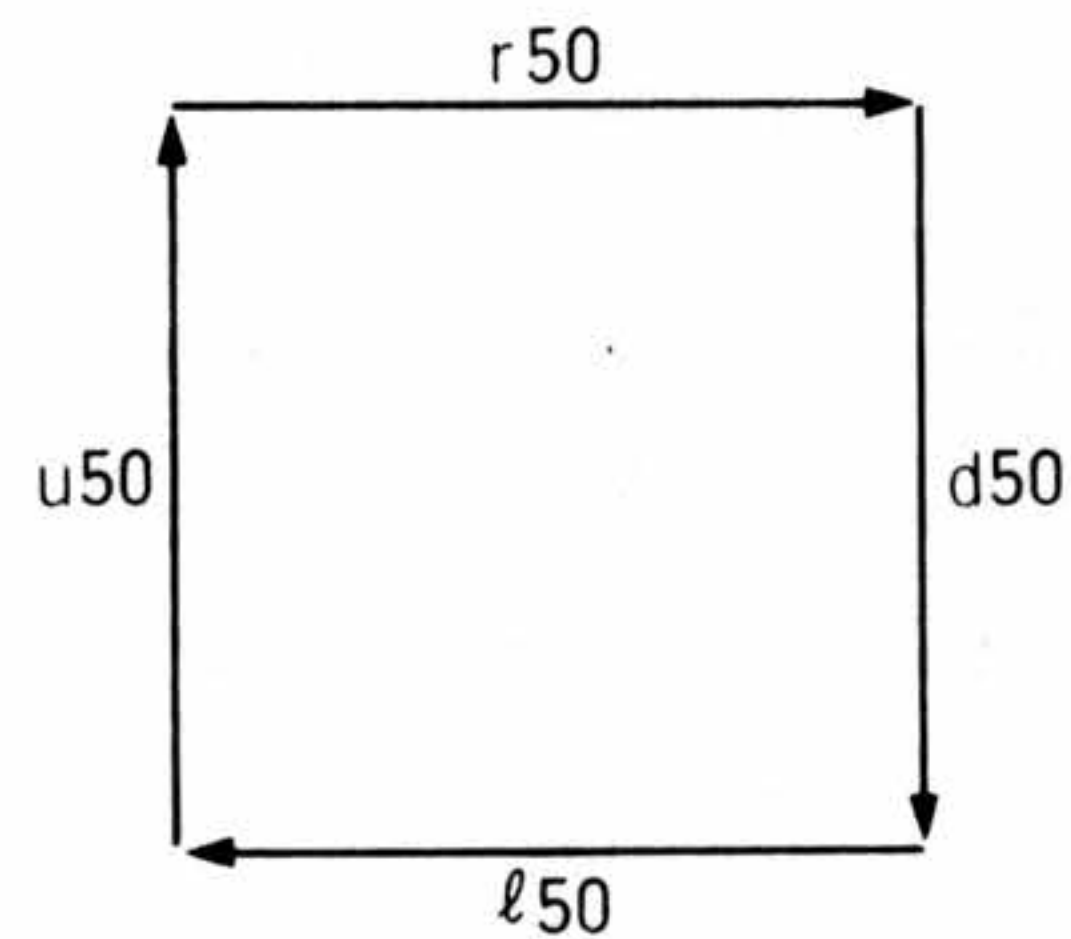
Un	上にn単位移動
Dn	下にn単位移動
Ln	左にn単位移動
Rn	右にn単位移動
En	右上にn単位移動
Fn	右下にn単位移動
Gn	左下にn単位移動
Hn	左上にn単位移動
Mx, y	指定した座標に移動
An	図形の角度を指定(n=0，1，2，3)
Cn	色を指定　$0 \leqq n \leqq 15$
Sn	スケール指定　$0 < n < 255$

15. 絵の描き方

たとえば，次の DRAW 命令は 1 辺が50ドットの四角形を描きます．

```
DRAW "R50 D50 L50 U50"
```

次のプログラムを実行すると，四角形が色を変えながら移動していきます．

```
10 SCREEN 2
20 FOR X=0 TO 150 STEP 10
30 DRAW "BM =X;, =X;"
40 C=X/10
50 DRAW "C=C;"
60 DRAW "R50 D50 L50 U50"
70 NEXT X
80 GO TO 80
```

30行を見てください．

```
30 DRAW "BM=X;, =X;"
```

ここで＝と；で囲まれた X は変数を表しています．このように，DRAW コマンドの中で変数を指定するときはその前後を＝と；で囲みます．
50行でも変数 C を使っています．
次のプログラムは，四角形の色と大きさを変えて描きます．

```
10 SCREEN 2
20 DRAW "BM 0, 0"
30 FOR S=1 TO 15
40 DRAW "S=S; C=S;"
50 DRAW "R50 D50 L50 U50"
60 NEXT S
70 GO TO 70
```

15. 絵の描き方

● 文字の表示

グラフィックモードでは，PRINT 文で文字を表示することはできません．文字を表示させたい場合は次のようにしてください．

```
OPEN "GRP:" FOR OUTPUT AS # <ファイル番号>
PRINT # <ファイル番号>, "文字列"
```

また，表示位置を指定したい場合には，LOCATE 文ではなく，PSET または PRESET 命令によって位置づけてください．

次のプログラムを実行すると，(20, 100) の位置から "MSX MICRO COMPUTER" と表示します．

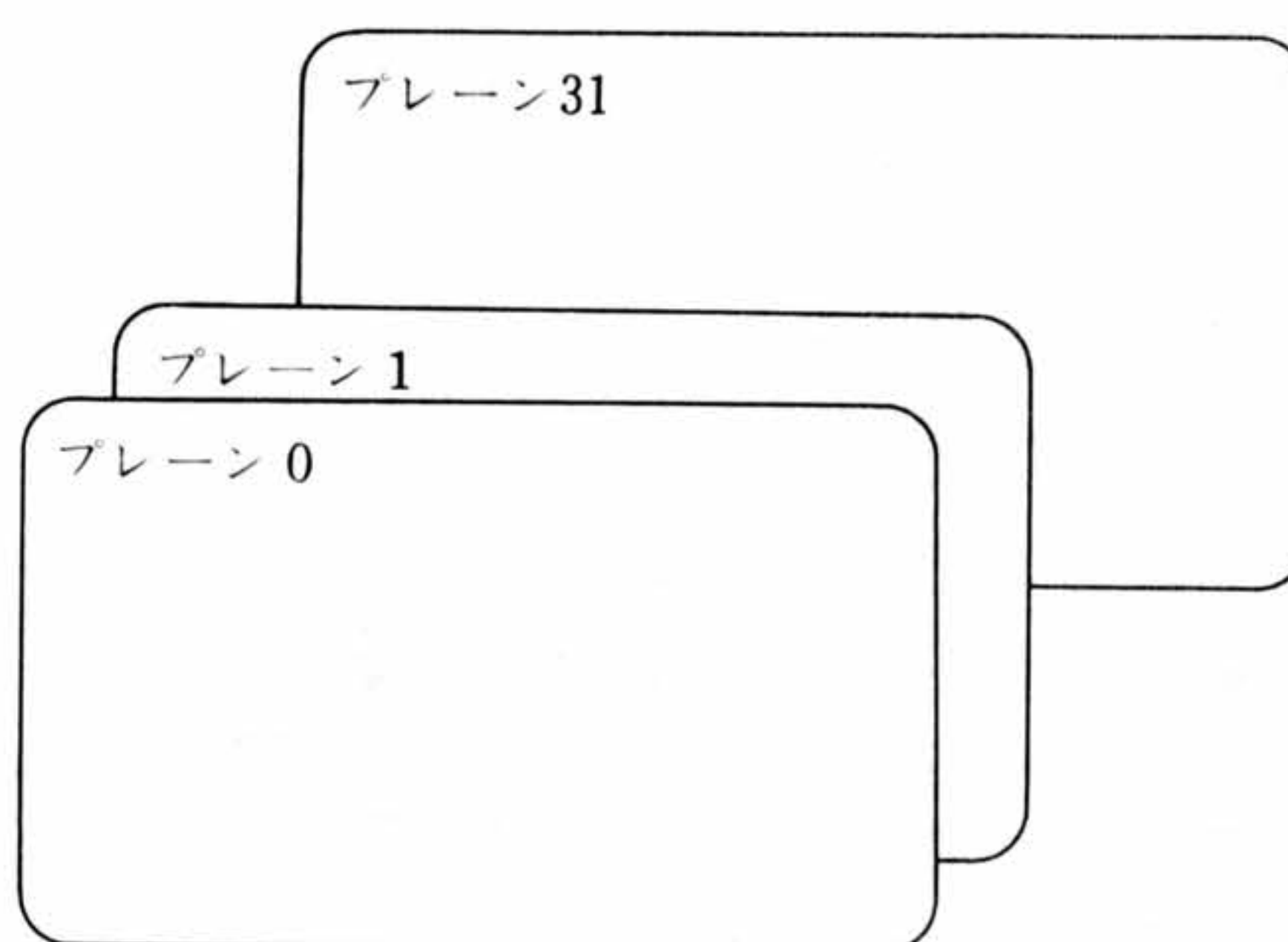
```
10 SCREEN 2
20 OPEN "GRP:" FOR OUTPUT AS #1
30 PRESET (20, 100)
40 PRINT #1, "MSX MICRO COMPUTER"
50 GOTO 50
```

16．絵の動かし方

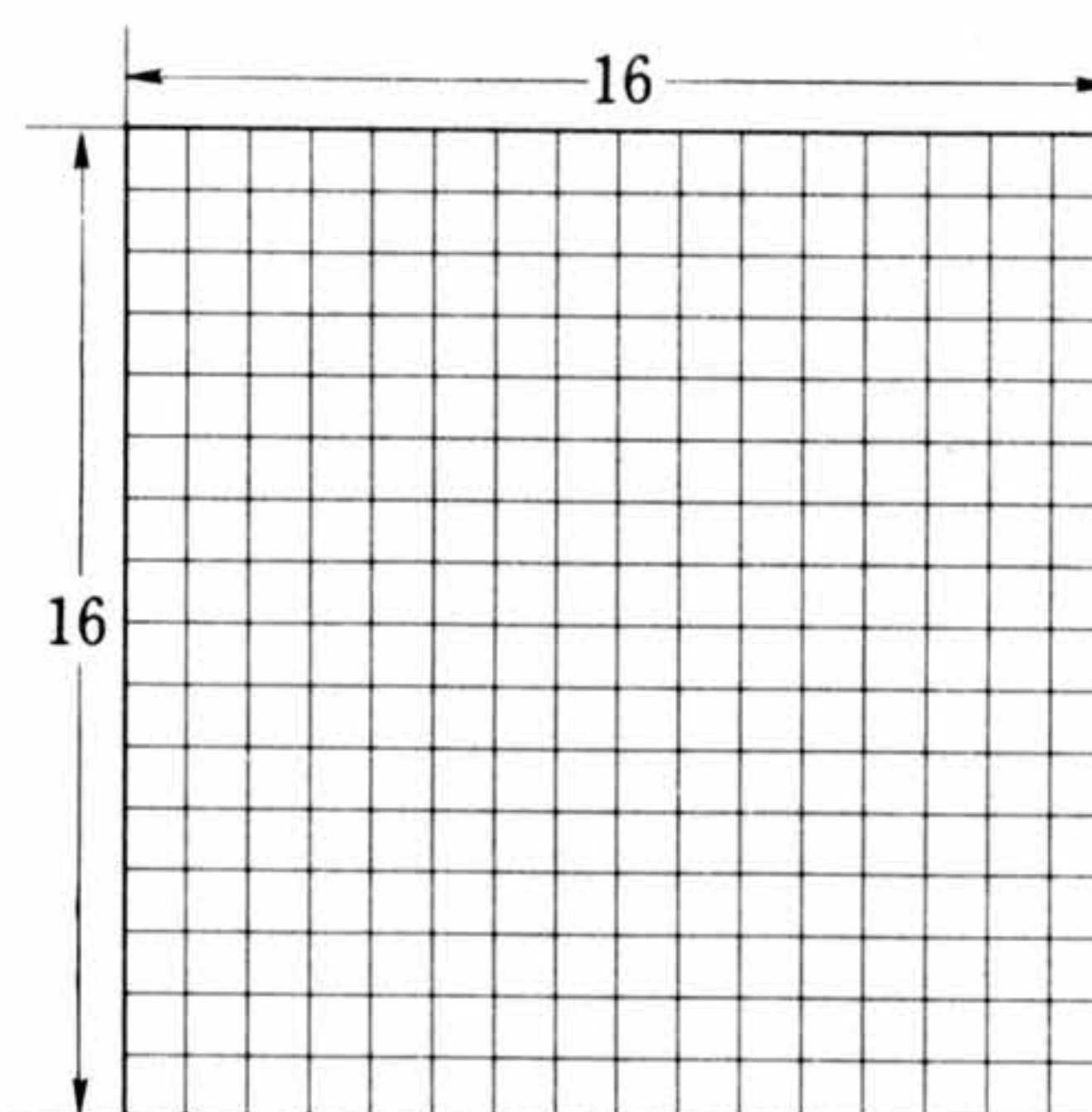
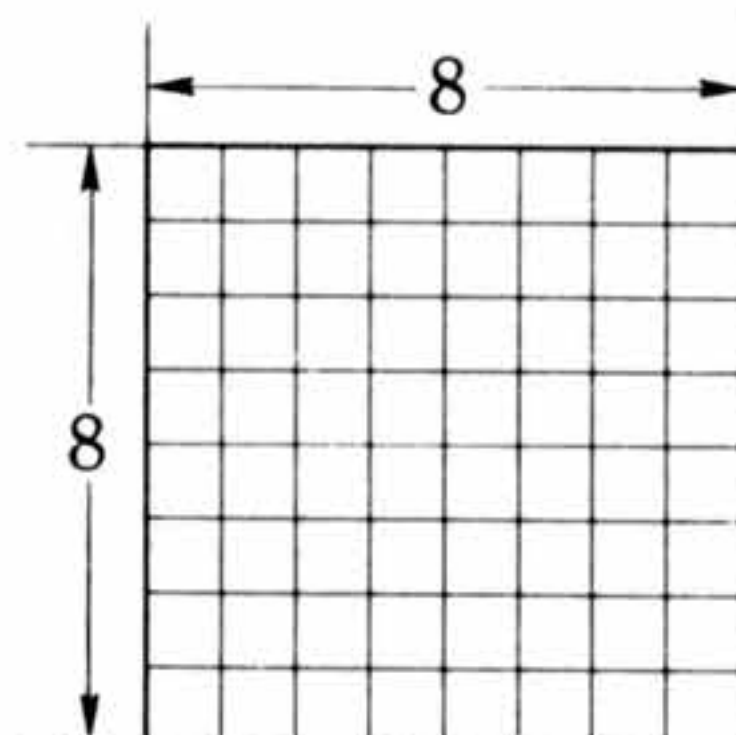
画面に描いた絵を動かすには，座標を変えながら PSET，PRESET を繰り返すことも一つの方法ですが，MSX BASIC ではもっと便利な「スプライト」という機能が用意されています．ここではスプライト機能について解説します．

スプライト

MSX では，32枚の仮想画面があり，それらをプレーンと呼んでいます．

仮想画面

プレーン

プレーンは32×24のテキストモードまたはグラフィックモードで使用することができます．

このプレーン１枚に，１つのスプライトを表示することができます．スプライトとは８×８ドットまたは16×16ドットで表されるパターンのことです．

16．絵の動かし方

このパターンのサイズはSCREEN命令で指定します．

SCREEN<モード>,<スプライトサイズ>

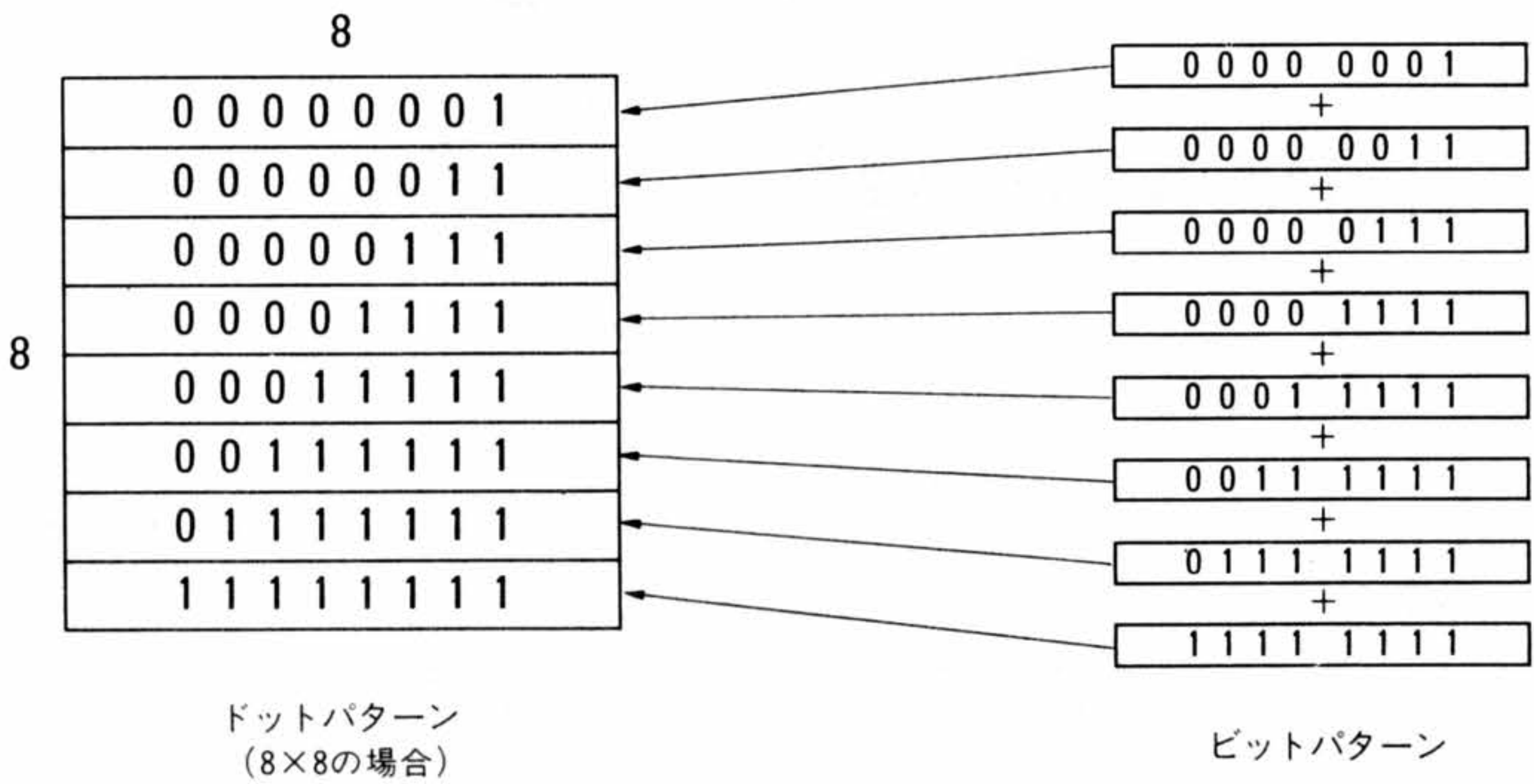
スプライトサイズ

0	8×8	
1	8×8	拡大
2	16×16	
3	16×16	拡大

スプライトサイズとして，1または3と指定すると，それぞれ2倍に拡大表示されます．

パターンの定義の仕方を説明しましょう．パターンの各ドットを1ビットで指定します．つまり，あるビットをオンにすると，対応するドットに点が打たれます．ビットとドットは8つずつで対応します．8ビットは1文字ですから，1文字で8ドットのパターンを指定できると考えても良いでしょう．

ビット P88

ドットパターン
（8×8の場合）

ビットパターン

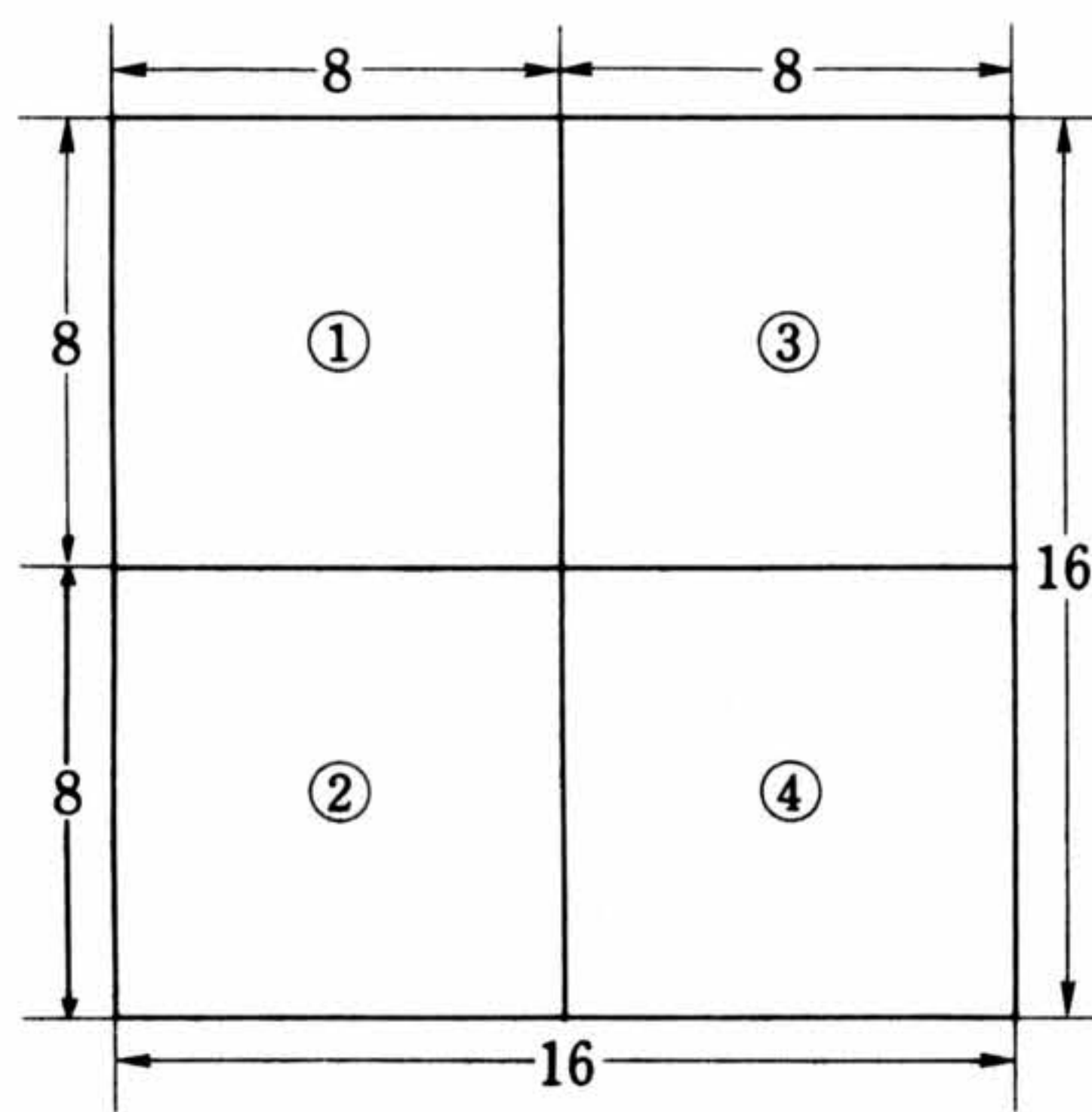
上図のパターンは

CHR$(1)+CHR$(3)+CHR$(7)+CHR$(&HF)+CHR$(&H1F)
+CHR$(&H3F)+CHR$(&H7F)+CHR$(&HFF)

で指定することができます．

16．絵の動かし方

16×16のドットパターンの場合は，８×８ドットパターンの定義を次の順に連結すれば良いのです．

〈①の定義〉＋〈②の定義〉＋〈③の定義〉＋〈④の定義〉

したがって16×16のパターンを定義するには，最大32文字必要となります．

定義したパターンは

SPRITE$(n)

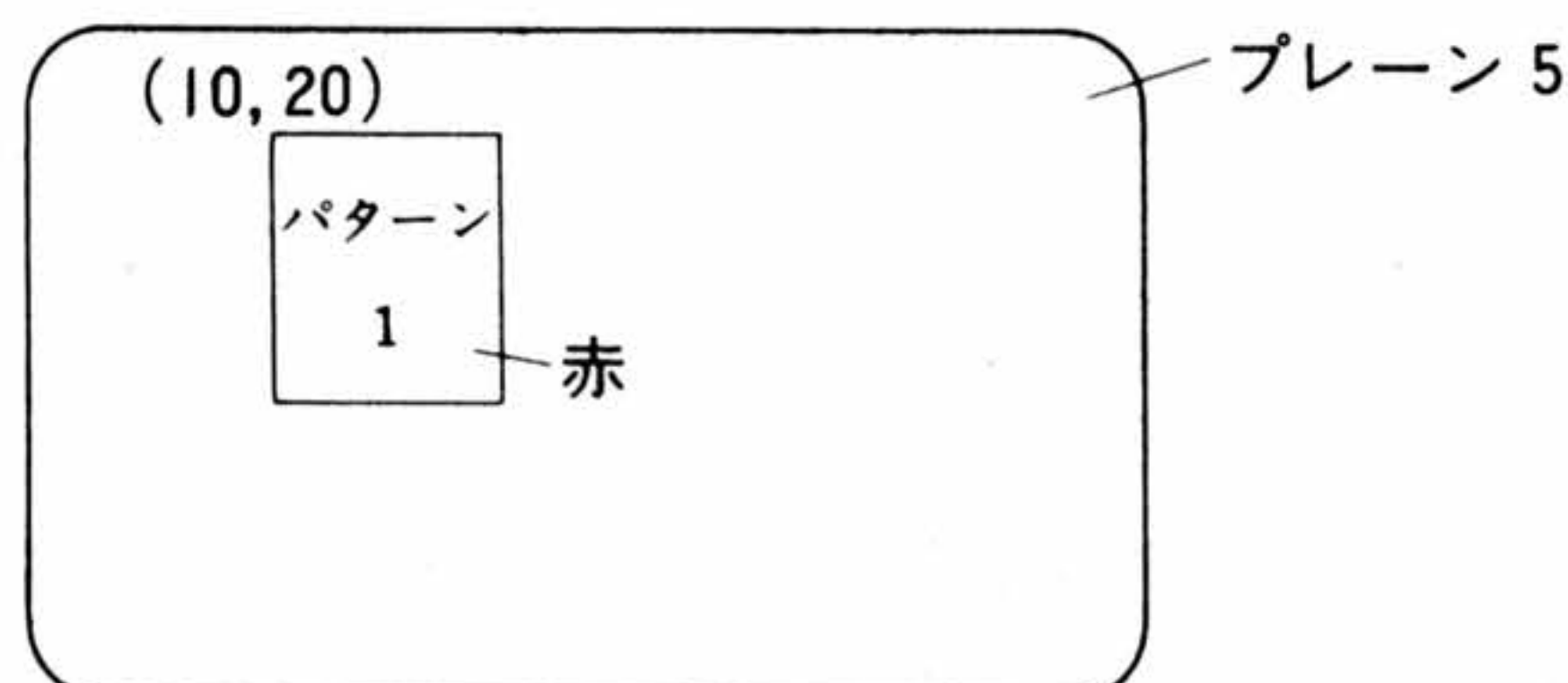
というシステム配列変数に代入しておかなければなりません．SPRITE$(n)に代入したパターンをパターンｎと呼びます．なお，８×８の最大パターン番号は255，16×16の最大パターン番号は63です．

登録したパターンをプレーンに書き込むにはPUT SPRITE命令を使います．

PUT SPRITE 〈プレーン番号〉，(X，Y)，〈色〉，〈パターン番号〉

たとえば

```
PUT SPRITE 5, (10, 20), 8, 1
```

とすると，プレーン５の座標（10，20）にスプライトパターン１を赤で表示します．

なお，スプライトパターンは１プレーンに１つしか書けません．また，プレーン全体で同時に書き込めるパターンの数は最大32です．

16. 絵の動かし方

次のプログラムは，3つのスプライトパターン0，1，2を定義し，それぞれプレーン0，1，2に表示するものです.

```
10 DIM S$(10):DEFINT A-Z
20 FOR S=0 TO 2:S$=""
30 FOR R=0 TO 7:READ T$
40 S$=S$+CHR$(VAL("&B"+T$))
50 NEXT
60 S$(S)=S$
70 NEXT
80 SCREEN 1,1
90 FOR S=0 TO 2:SPRITE$(S)=S$(S):NEXT
100 DATA 00011000
110 DATA 00100100
120 DATA 01000010
130 DATA 10000001    スプライト0の定義
140 DATA 01000010
150 DATA 00100100
160 DATA 00011000
170 DATA 00000000
180 DATA 00011000
190 DATA 00011000
200 DATA 00011000
210 DATA 11111111    スプライト1の定義
220 DATA 11111111
230 DATA 00011000
240 DATA 00011000
250 DATA 00011000
260 DATA 11111111
270 DATA 01111110
280 DATA 00111100
```

```
290 DATA 00011000
300 DATA 00111100
310 DATA 01111110
320 DATA 11111111
330 DATA 11111111
340 PUT SPRITE 0,(50,50),9
350 PUT SPRITE 1,(100,100),7
360 PUT SPRITE 2,(150,155),3
```

（290〜330：スプライト2の定義）
（340：……スプライト0をプレーン0に書き込む．）
（350：……スプライト1をプレーン1に書き込む．）
（360：……スプライト2をプレーン2に書き込む．）

割り込み

プレーンのスプライトが重なりあったときに，割込みをかけることもできます．それには次の2つの命令を使います．

```
SPRITE ON／OFF／STOP
```

```
ON SPRITE GOSUB <行番号>
```

SPRITE ON とすると，スプライトが重なりあったとき割込みが発生し，そのとき，ON SPRITE GOSUB で指定したサブルーチンが実行されます．

サブルーチン

乱数

次のプログラムは，先ほど定義したパターンを乱数を使って動かし，重なりあったとき，画面を点滅させるものです．

```
10 DIM S$(10):DEFINT A-Z
20 FOR S=0 TO 2:S$="".
30 FOR R=0 TO 7:READ T$
40 S$=S$+CHR$(VAL("&B"+T$))
50 NEXT
60 S$(S)=S$
70 NEXT
80 SCREEN 1,1
90 FOR S=0 TO 2:SPRITE$(S)=S$(S):NEXT
100 DATA 00011000
110 DATA 00100100
120 DATA 01000010
130 DATA 10000001
140 DATA 01000010
150 DATA 00100100
160 DATA 00011000
170 DATA 00000000
180 DATA 00011000
190 DATA 00011000
200 DATA 00011000
```

16．絵の動かし方

```
210 DATA 11111111
220 DATA 11111111
230 DATA 00011000
240 DATA 00011000
250 DATA 00011000
260 DATA 11111111
270 DATA 01111110
280 DATA 00111100
290 DATA 00011000
300 DATA 00111100
310 DATA 01111110
320 DATA 11111111
330 DATA 11111111
340 SPRITE ON
350 ON SPRITE GOSUB 450   ……割込みルーチンの指定
360 D=RND(-TIME)                        ┐
370 X0=RND(1)*240:Y0=RND(1)*180         │
380 X1=RND(1)*240:Y1=RND(1)*180         │ 乱数発生
390 X2=RND(1)*240:Y2=RND(1)*180         ┘
400 PUT SPRITE 0,(X0,Y0),9   ┐
410 PUT SPRITE 1,(X1,Y1),7   │ スプライトの書き込み
420 PUT SPRITE 2,(X2,Y2),3   ┘
430 FOR I=1 TO 200:NEXT I
440 GOTO 370
450 SPRITE OFF               ┐
460 FOR I=1 TO 10:BEEP       │
470 COLOR 15,1,15            │
480 FOR J=1 TO 100:NEXT J    │
490 COLOR 15,15,1            │ スプライトパターンが重なると
500 NEXT I                   │ このルーチンが実行される.
510 COLOR 15,4,5             │
520 PUT SPRITE 0,(0,208)     │
530 SPRITE ON                ┘
540 RETURN
```

17．文字で絵を作る

テキストモードでも文字を使えば絵を描くことができます．

次のプログラムを見てください．

```
10 N=1
20 FOR S=1 TO N
30 PRINT "*";
40 NEXT S
50 PRINT
60 N=N+1
70 IF N>21 THEN END
80 GOTO 20
```

このプログラムを実行すると，画面１枚に次のような三角形をアスタリスク（*）を使って描きます．

```
*
**
***
****
*****
******
*******
********
*********
**********
***********
************
*************
**************
***************
****************
*****************
******************
*******************
********************
*********************
```

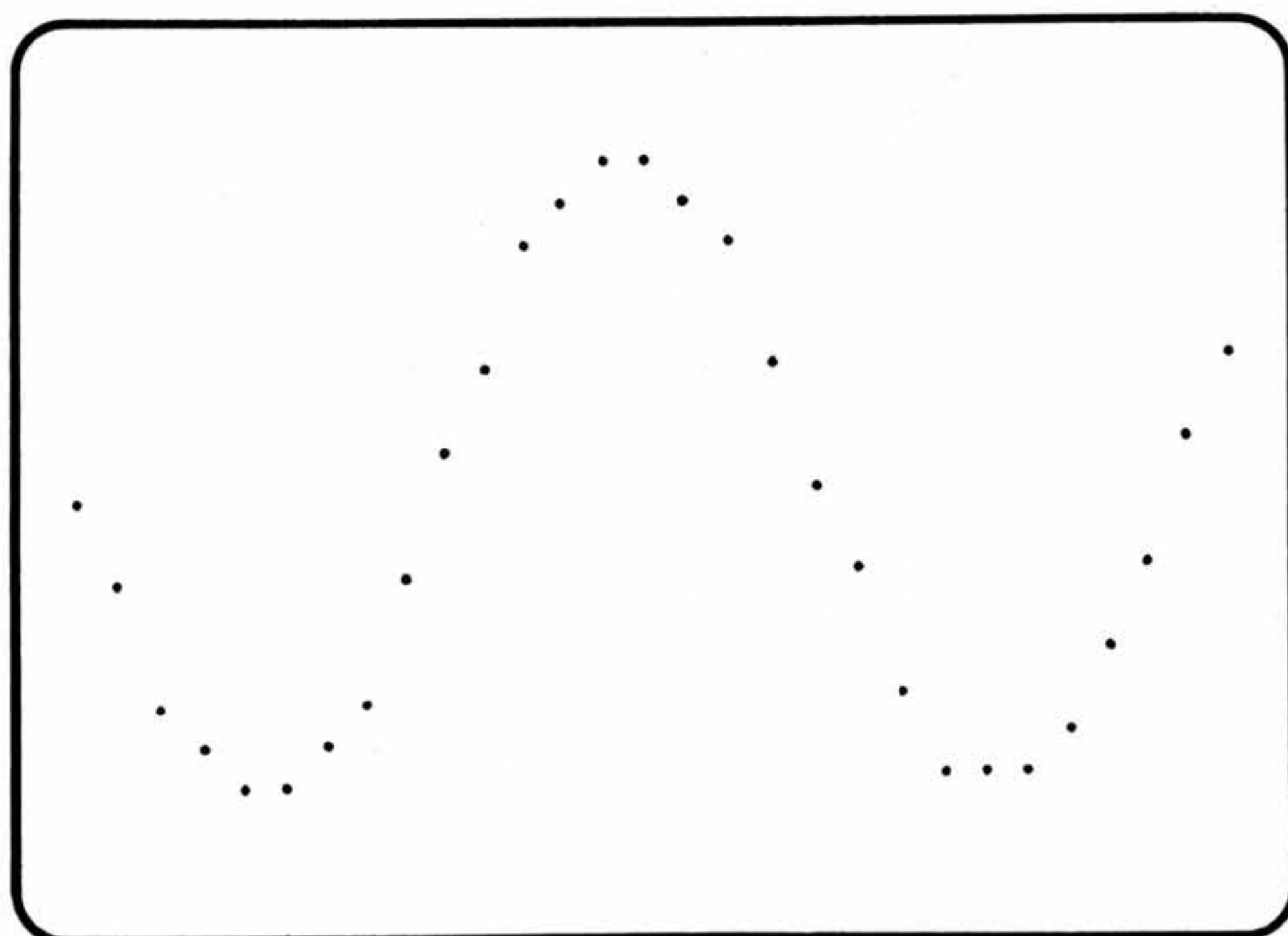
次のようにSTRING$関数を使うともっと簡単に描くことができます．

```
10 FOR I=1 TO 21
20 PRINT STRING$ (I, "*")
30 NEXT I
```

17. 文字で絵を作る

このプログラムを実行すると先ほどとまったく同じ三角形を＊を使って描きます.

次に LOCATE 文を使った例を示しましょう. 次のプログラムを実行すると, ピリオド（.）で SIN 曲線を描きます.

```
10 CLS
20 FOR X=0 TO 28
30 Y=SIN (T) * 8+12
40 LOCATE X, Y : PRINT 〝.〟;
50 T=T+.35
60 NEXT X
70 LOCATE 0, 0
```

棒グラフを描くこともできます. 次のプログラムは, DATA 文からデータを読み込んで, 成績を棒グラフで示すものです.

```
10 CLS
20 FOR K=1 TO 9
30 READ N$, D
40 PRINT N$ ; TAB (10) ;
50 FOR I=1 TO D
60 PRINT 〝●〟;
70 NEXT I
80 PRINT
90 NEXT K
```

17. 文字で絵を作る

```
100 DATA aoki, 5
110 DATA saitou, 10
120 DATA kimura, 8
130 DATA esaki, 7
140 DATA toda, 9
150 DATA kato, 2
160 DATA sugata, 1
170 DATA yamamoto, 6
180 DATA ueno, 3
```

成績は10点満点です．グラフィック記号●を使ってグラフを描きます．では実行してみましょう．

このプログラムも，もっと簡単にすることができます．

```
10 CLS
20 D$=STRING$ (10, "●")
30 FOR K=1 TO 9
40 READ N$, D
50 PRINT N$ ; TAB (10) ;
60 PRINT LEFT$ (D$, D)
70 NEXT K
100 DATA aoki, 5
      ⋮
以下同じ（データ文）
```

17. 文字で絵を作る

あるいは

```
10 CLS
20 FOR K=1 TO 9
30 READ N$, D
40 PRINT N$ ; TAB (10) ;
50 PRINT STRING$ (D, "●")
60 NEXT K
100 DATA aoki, 5
    ⋮
以下同じ（データ文）
```

18. キーボードを読む命令

● STOP キー

実行

プログラムの実行が始まってしまうと，特にプログラム中にキーボードの状態を見に行けという命令がない限り，キーボードはまったく無視されます．たとえば

```
10 PRINT "!";
20 GOTO 10
```

というプログラムを作ってRUNしてしまうと，もうキーボードから何を命令しても受けつけず，"！"の文字をPRINTし続けます．この場合でも，STOPキーだけは有効です．STOPキーを押すと実行は一時停止します．しかしそれはただ停止しただけで，他のキーはあいからず無視されています．ここでもう一度STOPキーを押すとまた実行を再開します．これで，STOPキーは，一時停止とその解除に使われることがわかりました．

次はCTRLキーを押しながらSTOPキーを押す場合です．"Break in 10"または"Break in 20"と"Ok"が表示され，プログラムの実行は停止してダイレクトモードになりました．今度はすべてのキーが有効になります．プログラムの実行を強制的に止めるにはCTRL＋STOPを押します．このとき"CONT"というコマンドを実行すると，またプログラムの続きを実行します．

ダイレクトモード
P 10

● INPUT

プログラム中でキーボードを見張る命令にはINPUT文があります．

```
10 INPUT A
20 PRINT A * 3.14
30 GOTO 10
```

INPUT文はすぐ後の変数にキーボードから値を代入する命令です．上のプログラムなら，10行が実行されたときに画面に"？"が現れます．次にキーボードから1 2 RETURNとキーを押すとAに12が代入され，プログラムの実行は20行に移ります．このとき，Aは数値変数ですから，A B

数値型
P 33

18. キーボードを読む命令

[RETURN] のように数値以外の文字を入力すると〝Redo from start〞というメッセージがでます．型が合わないのでやり直せという意味です．文字を入力したい場合は

```
10 INPUT A$
```

のように文字型の変数を用意します．

INPUT 文は，何の入力をしているのかがわかるように，プロンプト文というものを出力させることができます．

プロンプト

```
10 INPUT "チョッケイ"; A
20 PRINT "エンシュウ="; A * 3.14
30 GOTO 10
```

こうすると，実行したときに

チョッケイ？ □
カーソル

のようになるので便利です．

また一度に複数の変数に値を代入するなら

```
10 INPUT "XY"; X, Y
```

のように各変数をカンマ〝,〞で区切ってやります．

INPUT 文の入力は常に [RETURN] キーによって終了するのですが，一度に複数の値を入力するときは，[5][,][6][RETURN] のようにカンマで区切ってやらねばなりません．

たとえばこの例のようにXとYの両方を一度に入力することを要求されているときに [5][RETURN] とすると，〝？？〞と出力されます．これは，まだ入力が足りないという意味ですから，Yの値も [6][RETURN] としてやれば，無事 INPUT 命令が終了します．

反対に2つの値しか要求されないのに [5][,][6][,][7][,][8][RETURN] などとすると〝Extra ignored〞と表示され，この場合の7や8は無視されたことになります．

18．キーボードを読む命令

● LINE INPUT

文字型
P 33

この命令はINPUT文とよく似ています．違うところは，？が出力されずに代入する変数は文字型だけであるということ，一度に複数の変数に代入することができないということです．簡単に言ってしまえば，カンマ（,）や引用符（"）を含むような文字列を変数に代入するためのコマンドです．

たとえば，"A，B" という文字列をA$に代入するとき

```
10 INPUT A$
20 PRINT A$
```

これでは次のようになってしまいます．

```
RUN
? A，B
? Extra ignored
A
Ok
```

Aという文字だけがA$に代入され，他は無視されてしまいました．

次にLINE INPUT文を使うと

```
10 LINE INPUT A$
20 PPINT A$
RUN
"A，B"
"A，B"
Ok
```

RETURN キーを押すまでに入力したキーはすべて代入されています．

● INPUT$

INPUT$はキーボードの文字を読むための関数で，何文字読むかを指定すれば，STOP キーやモードキー以外のすべてのキーを読みとることができます．たとえばA$の中に，RETURN や CLS など文字にならないコードを代入するには，A$=CHR$ (13)+CHR$ (12) のようにする方法があります．これをキーボードから代入するには次のようにします．

18．キーボードを読む命令

```
10 A$=INPUT$ (2)
```

これが実行されると，STOP キー以外なら CTRL ＋ B （CHR$ (2) に対応します）とか → （CHR$ (28)）なども含めて，とにかく2文字押されるまで待ち，A$にそれらを代入します．このとき INPUT 文などと違って〝?″も，押した文字自体も画面には現われません．

それでは実際に何かやってみましょう．

```
10 CLS
20 A$=INPUT$ (3)
30 FOR I=1 TO 5
40 PRINT A$ ;
50 NEXT
```

RUN したら， → ， ↓ ， A キーを押してみてください．画面に， → ↓ A という動作を5回行ったのと同じ結果が現れるはずです．

またINPUT$のおもしろい使い方として，デバッグ中などにプログラムを1ステップずつ行わせるというようなことがあります．

デバッグ P 112
ステップ

```
10 SCREEN 2
20 FOR I=0 TO 15
30 LINE (50, 50)− (100, 100), I, BF
40 A$=INPUT$ (1)
50 NEXT
```

Iのカラーコードで画面に四角を描くのですが，何かキーを1つ押すごとにその色が変わります．40行をとってしまうと実行が速すぎて，その様子がわからなくなってしまいます．

カラーコード P 56

● INKEY$

INKEY$もキーボードの文字を読む関数ですが，INPUT$と違って読む文字数は指定できません．常に1文字を読むだけです．しかしこの関数が呼び出されたときに何もキーが押されていないと，〝何も押されていない″という答え，つまりヌルストリング（〝″ または CHR$ (0)）を返します．次のプログラムを実行してみてください．

ヌルストリング

18. キーボードを読む命令

```
10 A$=INKEY$
20 IF A$= "" THEN A$= "なにかキーをおして！"
30 PRINT A$
40 FOR I=0 TO 100 : NEXT
50 GOTO 10
```

タイマー

40行は実行速度を遅くするためのもので，タイマーと呼ばれます．キーを何も押さないとA$は〝″（ヌルストリング）になり20行で〝なにかキーをおして！″に変えられている様子がわかると思います．

何もキーを押していない，といっても実は「キーバッファがからである」という意味です．次のプログラムを実行してみるとよくわかります．

```
10 FOR I=0 TO  10000 : NEXT
20 A$=INKEY$
30 N$=N$+A$ : PRINT N$
40 IF A$<> "" THEN 20
50 PRINT N$
```

RUNしたら，できるだけ速く[A]のキーを40回〜50回押してください（10行のループが終わるまでの十数秒の間に！）．すると20行の実行が始まり，もうキーを押すのはやめたはずなのに，しばらくはN$がどんどん大きくなりながら表示されます（30行）．そしてN$の長さが39文字になったところで終わるはずです．つまり，INKEY$に文字を読ませるには，その文字が39字以内なら先に行入力することもできるというわけです．このように先に押されたキーは一度覚えておく所があり，これをキーバッファと呼びます．実は，これまでにでてきたINPUT文，INPUT$関数も同様で，更にプログラム実行終了後にも生きていることがあります．

キーバッファ

```
10 FOR I=0 TO 10000 : NEXT
20 A$=INPUT$(3)
30 PRINT A$
40 END
```

RUNしたらすぐに10秒位の間だけできるだけ多くのキーをいろいろ押してみてください．

18．キーボードを読む命令

```
RUN
ljh
Ok
lfdjknolfd lfdjkim □
```

Okが出た後，キーバッファにたまったものがちょうどダイレクトモードになってからキーを押したのと同じように表示されます．キーバッファにはもちろん RETURN やカーソルキーもたまっていきますから，たとえばINPUT文が実行されたときなどにこれが表示されるとおかしなことになることがあります．キーバッファをからにするには

```
100 IF INKEY$<> "" THEN 100
```

のような行を挿入しておけばよいでしょう．

また，気がついた人もいると思いますが，キーバッファに文字がたまっていくときは，１つのキーをいくら押しっぱなしにしてもオートリピートはきかず，１文字しかたまりません．またバッファが39文字いっぱいになるといくらキーを押してもキークリックの音がしなくなるので，キーを受けつけていないことがわかります．

キークリック

最後にINKEY$の実用例として，カーソルキーで画面上の点を上下左右に動かすプログラムを紹介します．いろいろなものに応用してみてください．

```
10 SCREEN 2 : COLOR 15, 4, 7 : CLS
20 X=128 : Y=96
30 PSET (X, Y) : PRESET (X, Y)
40 A$=INKEY$
50 IF A$=CHR$ (28)  AND×<255 THEN X=X+1
60 IF A$=CHR$ (29) ANDX>0 THEN X=X-1
70 IF A$=CHR$ (30)  AND Y>0 THEN Y=Y-1
80 IF A$=CHR$ (31) AND Y<191 THEN Y=Y+1
90 GOTO 30
```

18. キーボードを読む命令

● INP関数

これまでの命令や関数は，キーバッファを対象とした読み取りで時間的な差があります．またあるキーを押しっぱなしにしても思うような結果が得られない，2つ以上のキーを同時に判定できないなどの問題があります．これはオートリピートの影響です．

オート
リピート
P(9)

INP関数は使い方は少し難しいのですが，キーバッファとは関係がなく，この関数が呼ばれたときに本当にキーが押されているかどうかだけを判定したり，複数のキーが押されているということを知ることもできます．また，キーボード以外の入力も検出することができますが，それについてはここでは解説しません．
それでは，まずは，使ってみることにします．

```
10 OUT 170, (INP (170) AND 240) OR 0
20 PRINT (INP (169) XOR 255) : GOTO 10
```

このプログラムを実行すると0が画面に出力され続けます．そして次の8つのキーを押すとその数字が変化します．

キー	7	6	5	4	3	2	1	0	何も押さない
	↓	↓	↓	↓	↓	↓	↓	↓	↓
数値	128	64	32	16	8	4	2	1	0

このとき，例えば[1]と[2]のキーを同時に押していればその値は2＋4で6に，[3]と[4]と[6]なら8＋16＋64で88のようになるので，複数のキーも判定できます．

その他のキーも8つづつの組になっていて10行の後のOR　0のところをOR 1，OR 2のように変えることによって対象となる組が変わります．

次の表を見てください．

18. キーボードを読む命令

返される値 / ORする値	128	64	32	16	8	4	2	1	
0	7	6	5	4	3	2	1	0	各
1	;	[	@	¥	^	−	9	8	
2	B	A	_	/	.	,	]	:	キ
3	J	I	H	G	F	E	D	C	
4	R	Q	P	O	N	M	L	K	ー
5	Z	Y	X	W	V	U	T	S	
6	F3	F2	F1	かな	CAPS	GRPH	CTRL	SHIFT	
7	RETURN	SELECT	BS	STOP	TAB	ESC	F5	F4	
8	→	↓	↑	←	DEL	INS	HOME	SPACE	

このプログラムを応用して20行の1部を

IF (INP (169) XOR 255)=……THEN……

のようにして何かを実行させることができます．10行のOUT命令と，20行のINP関数を使う命令の間が離れていると，キーの判定ができません．できればマルチステートメントにしてしまった方がよいと思います．

マルチステートメント P16

OUT…………：IF……INP (169) XOR 255=……

このとき（INP (169) XOR 255）とするのは返される値をわかり易くするためのもので，単に INP (169) とすることもできます．この場合返される数値は次のようになります．

INP (169) XOR 255	128	64	32	16	8	4	2	1	0
INP (169)	127	191	223	239	247	251	253	254	255

複数のキーを判定するなら2進数で考えた方が簡単です．

2進数 P89

18. キーボードを読む命令

簡単な応用例を掲げておきます.

```
10 SCREEN 2,0 : X=128 : Y=96
20 SPRITE$ (0)= "AcUIIUcA"
30 OUT 170, (INP (170) AND 240) OR 8
40 A=INP (169) XOR 255
50 IF A AND 16 THEN X=X-1
60 IF A AND 128 THEN X=X+1
70 IF A AND 32 THEN Y=Y-1
80 IF A AND 64 THEN Y=Y+1
90 PUT SPRITE 0, (X, Y) : GOTO 30
```

カーソルキーで図形を移動します. キーを押し続けたり, 2つ以上のキーを押したときに, INKEY$を使った場合に比べて思い通りに動かすことができます.

このプログラムを解析するときは,「論理演算」,「ビット／バイト」, の頃やSPRITE機能についても読んでおいた方がわかりやすいでしょう.

実用上においてはこれで十分ですが, INPの説明の初めの方でキーを押すと, 画面の0が1や4などに変わる実験中, キーを押したままなのに, たまに0がでてしまうことに気付いた人がいるかもしれません. これはハードウェアの制約で, このヌケを完全に防ぐためには機械語の力が必要になってきます.

ハードウェア
機械語 P 106

```
10 CLEAR 200, &HDFEF : AD=&HDFF0
20 FOR I=0 TO 8
30 READ A$ : POKE AD+I, VAL ("&H"+A$)
40 NEXT
50 DEFUSR=AD
60 DATA 3E,08,CD,41,01,32,F9,DF,C9
70 U=USR (0) : PRINT PEEK (AD+9) XOR 255 : GOTO 70
```

このプログラムを実行すると, カーソルキーなどを押し続けている間は0が返されることは絶対になくなります. その他のキーの組を判定するには60行の2番目のデータ08を00～07に変えてください.

19. 論理演算

1＋2＝3とか2×4＝8のような算術は，誰にでも簡単にできるものですが，1 AND 2=0 とか 2 OR 4=6 などコンピュータ言語で扱う論理演算は，初めのうちは理解しにくいものかもしれません．いったいどんなものなのでしょう．

まず，理屈ぬきでふだん何気なくプログラムの中で使っている論理演算をみてみます．

```
10 INPUT A
20 IF A<0 THEN PRINT "MINUS" : GOTO 10
30 IF A>0 THEN PRINT "PLUS" : GOTO 10
40 PRINT "ZERO" : GOTO 10
```

20行や30行の IF 文中には，A<0 とか，A>0 のような条件が書かれています．これは，それぞれもしAが0より小さければ，もしAが0より大きければ，という意味であることが，人間の言葉そのものです．ところが，コンピュータはこれを判定するとき，その式が真ならー1，偽なら0を返すことによって判断します．ダイレクトモードで実験してみます．

真偽

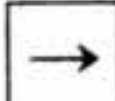
```
?2<3
-1
Ok
?2>3
 0
Ok
?5=5
-1
Ok
```

つまり，20行の IF 文なら

```
20 IF A<0 THEN PRINT "MINUS" : GOTO 10
```

A<0 の部分がー1ならば THEN 以下が実行されるのです．

そこで20行を，

```
20 IF -1 THEN PRINT "MINUS" : GOTO 10
```

と変えてプログラムを実行します．このとき，A にどんな値を代入しても，20行の THEN 以下は必ず実行され，MINUS がプリントされます．逆に，

```
20 IF 0 THEN……
```

とすると，A がマイナスの値でも，20行の THEN 以下は実行されません．

このような真と偽の値を利用して，プログラムを簡略化することができます．たとえば，画面上の点をカーソルキーで左右に動かすプログラムを考えてみます．

```
10 SCREEN 2 : X=128 : Y=50
20 PSET (X, Y) : PRESET (X, Y)
30 A$=INKEY$
40 IF A$=CHR$(28) AND X<255 THEN X=X+1
50 IF A$=CHR$(29) AND X>0 THEN X=X-1
60 GOTO 20
```

40行は［→］キーを押していて，しかも X が254以下なら（画面から点がはみださないように）X を右に 1 つずらして点を移動させるというものですが，論理演算を使えば次のように書けます．

```
40 X=X+ (A$=CHR$ (28)) * (X<255)
```

同様に50行も

```
50 X=X- (A$=CHR$ (29)) * (X>0)
        └──────────────┘   └─────┘
  押されたキー(A$)が          X>0なら
  CHR$(29)(←)なら真(-1)      真(-1)
```

どちらかでも成り立たないと

```
X=X-(0) * (-1) または X=X-(-1) * (0)   または X=X-(0) * (0)
    └────────┘             └────────┘              └───────┘
        0                      0                       0
```

両方成り立つと

```
X=X-(-1) * (-1)
    └──────────┘
         1
```

更に，40行と50行をつなげることもできます．

```
40  X=X+(A$=CHR$(28)) * (X<255)-(A$=CHR$(29)) * (X>0)
        └─────────────────────┘ └───────────────────┘
                   1                   0  →Xは+1される
                   0                   1  →Xは-1される
                   0                   0  →Xは変わらない
```

19. 論理演算

これをIF文でやると大変です.

```
40 IF A$=CHR$ (28) AND X<255 THEN X=X+1 ELSE IF A$=
CHR$ (29) AND X>0 THEN X=X−1
```

このように, X<5とか2＝2のような式の真偽は, 常に0か−1で返されることを知っておいてください.

さて, IF文の次に0や−1以外がきたときはどうなるのでしょう. 次のプログラムを実行してみます.

```
10 FOR I=−5 TO 5
20 PRINT I ;
30 IF I THEN PRINT "TRUE" ELSE PRINT "FALSE"
40 NEXT
RUN
−5       TRUE
−4
 ⋮         ⋮
−1       TRUE
 0       FALSE
 1       TRUE
 ⋮         ⋮
 5       TRUE
Ok
```

このように, 0以外はすべて真になります. なぜこうしてあるのかを知るためには, 数値をすべて2進数にしてみる必要があります. なぜなら, コンピュータの内部では, 数は2進数で格納されるからです. MSX BASICでは16ケタまでの2進数を扱います.

2進数 P 89

数値は?BIN$（5）のようにして簡単に表示することができます.

5＝101　　2＝10　　0＝0

負の数は, 左側に仮の17ビット目があるものとして引き算して作ります. これを2の補数と呼びます.

2の補数

```
  ┌→仮の17ビットをたて
 0= 1 0000000000000000
      そこから1を引くと
−1= 0 1111111111111111        ?BIN$(−1)で表示されます.
      └───────────────┘
              答
      更に1を引くと
−2=1111111111111110
```

19. 論理演算

補数を求めるには次のようにすると簡単です.

たとえば−39なら

39の2進　　0000000000100111

ビットを反転　1111111111011000

1を足す　　1111111111011001　　?BIN$（−39）

ここでさきほど，偽は0，真はそれ以外だったことを思い出してください．つまり，コンピュータの判定は，16ケタの2進数の中で1（ビットが立っている）が1つもなければ偽（0）で，1つでもあれば真（0以外）としているのです．またx＜0が真のとき，−1を返すのにも理由があります．ここで論理演算のNOTを取りあげてみましょう．

次のプログラムを実行してみてください．

```
10 TRUE=-1
20 IF TRUE THEN PRINT "TRUE" ELSE PRINT "FALSE"
30 TRUE=NOT TRUE
40 GOTO 20
RUN
TRUE
FALSE
TRUE
FALSE
(CTRL-STOP を押すまで)
```

このように，真と偽を交換するには−1と0が最適なのです．NOTはすべてのビットを反転しますから，「真の反対は偽，偽の反対は真」とすることができます．

11　11　11　11　11　11　11　11

↓

NOT

00　00　00　00　00　00　00　00

↓

NOT

11　11　11　11　11　11　11　11

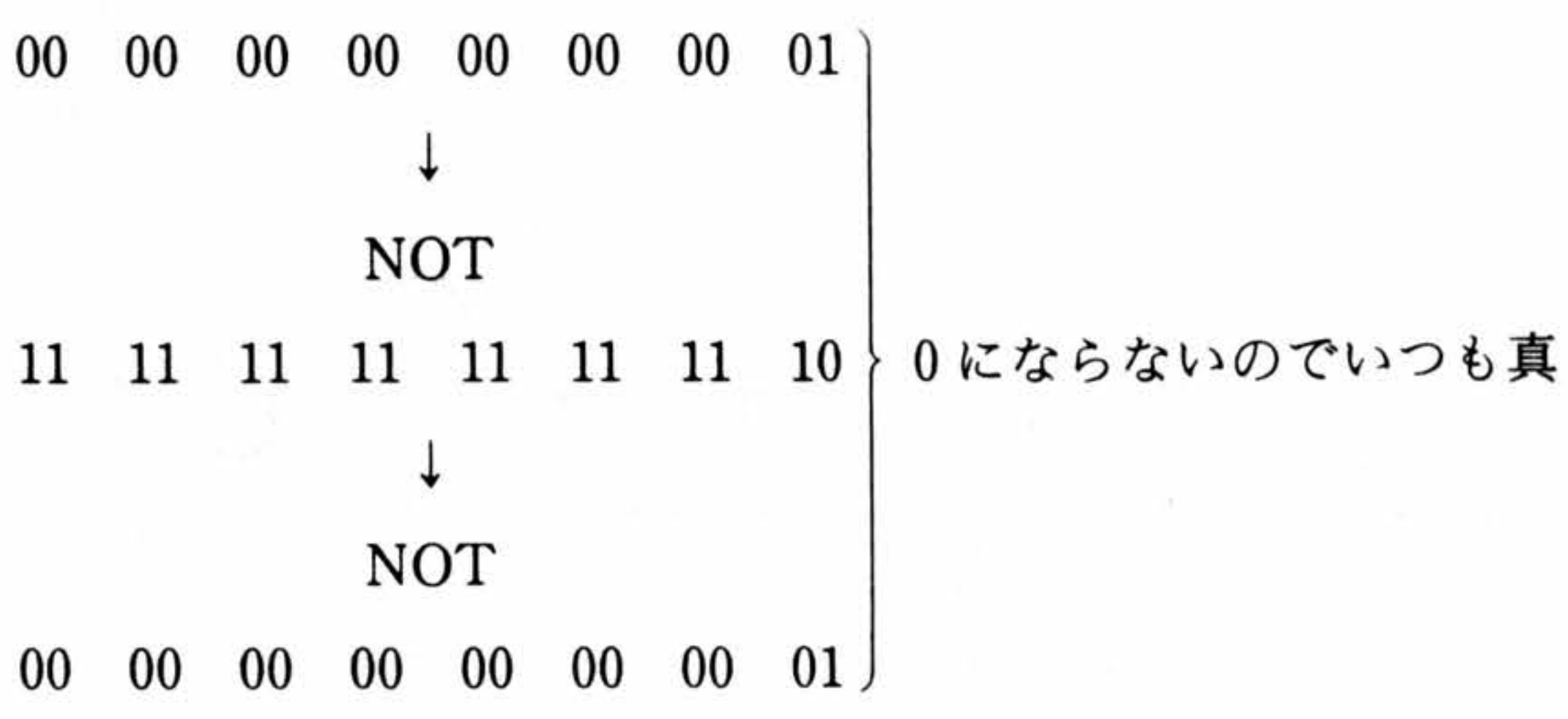
となります．

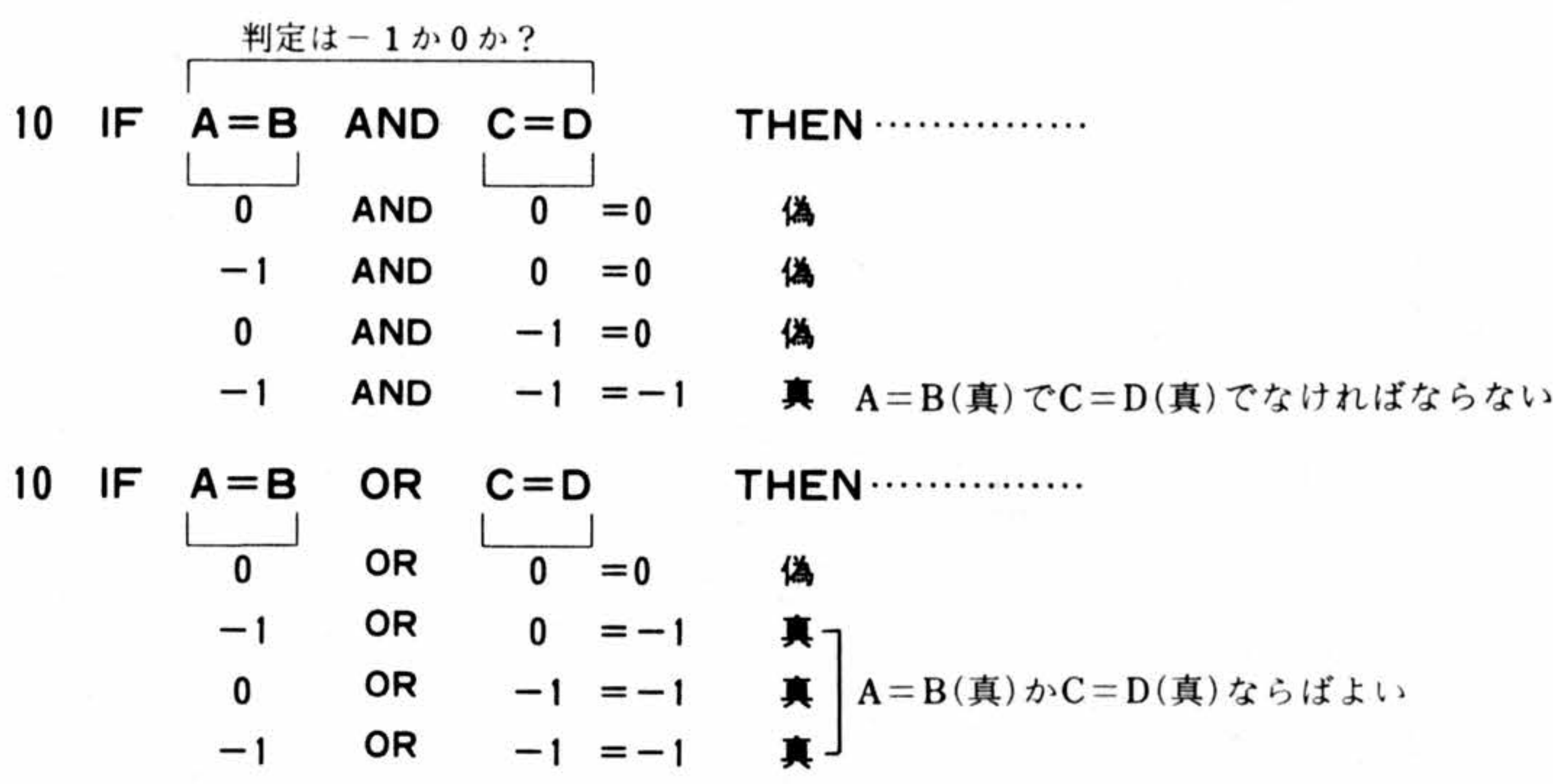
19. 論理演算

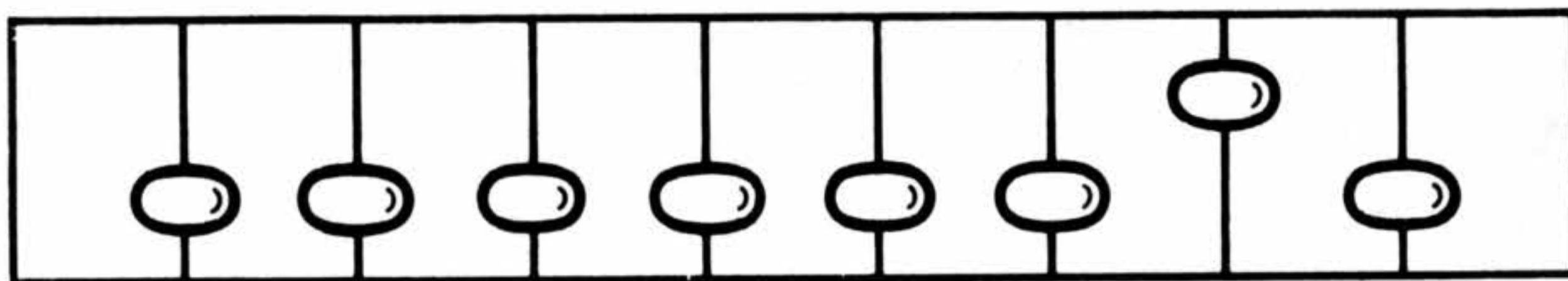
実際に10行を

```
10 TRUE=1
```

のようにしてしまうと，30行でNOTしてもTRUEという変数は－2で，やはり真になってしまい，FALSEの文字は現れません．

```
00 00 00 00 00 00 00 01 ┐
          ↓             │
         NOT            │
11 11 11 11 11 11 11 10 ├ 0にならないのでいつも真
          ↓             │
         NOT            │
00 00 00 00 00 00 00 01 ┘
```

ここまでわかれば，その他のAND, OR, XORなどは，式と演算の項などを見ていただければよくわかるはずです．何気なく使っているIF文中のANDやORなども，実にうまく機能していることがわかるでしょう．

```
           判定は－1か0か？
10  IF  A=B  AND  C=D    THEN……………
         0   AND   0  =0     偽
        －1  AND   0  =0     偽
         0   AND  －1 =0     偽
        －1  AND  －1 =－1   真  A=B(真)でC=D(真)でなければならない
10  IF  A=B  OR   C=D    THEN……………
         0   OR    0  =0     偽
        －1  OR    0  =－1   真 ┐
         0   OR   －1 =－1   真 ├ A=B(真)かC=D(真)ならばよい
        －1  OR   －1 =－1   真 ┘
```

20. ビット/バイト

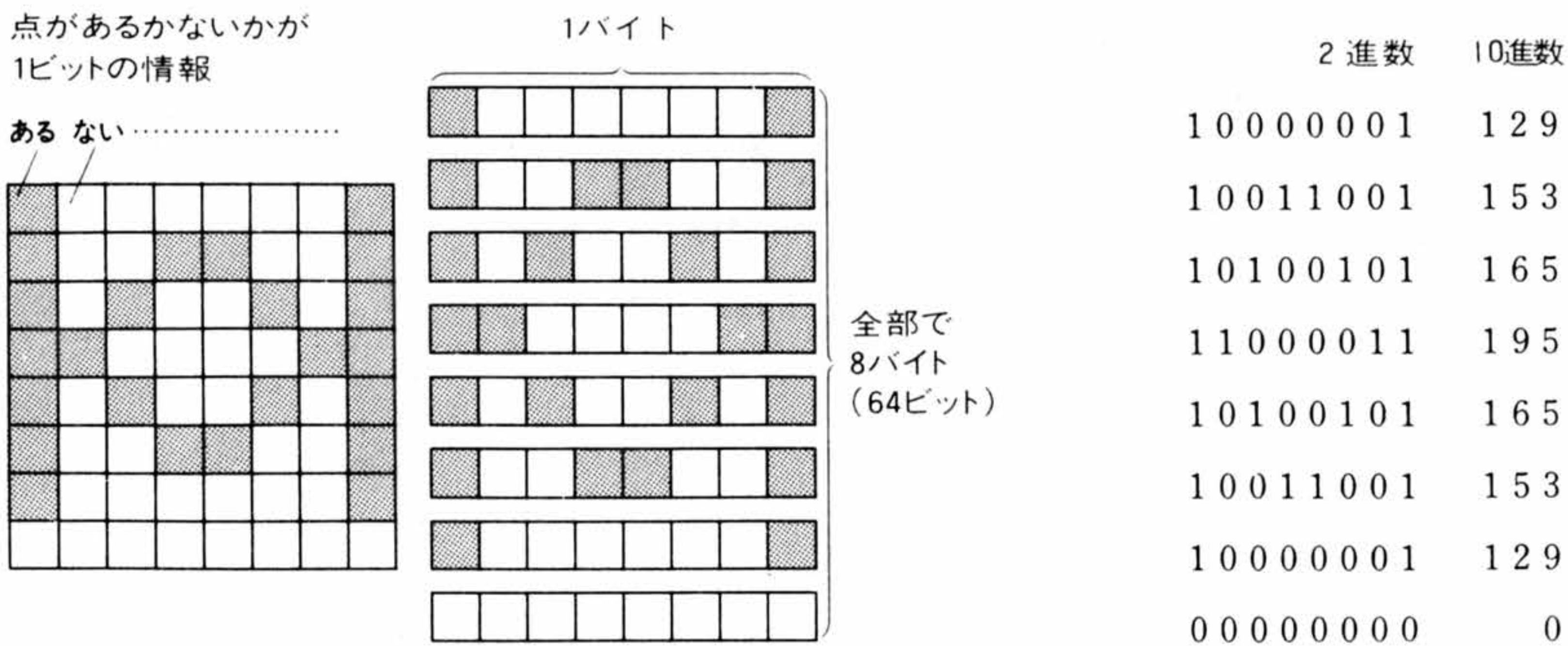
1ビット（1 bit）は1ケタの2進数で表現される情報です．簡単に言うと，あるかないか，上か下か，＋か－かなどは，すべて1ビットの情報で表すことができます．コンピュータの動作は，細かく分解していくと最後には必ずこの1ビットの単位になります．要するに，電気信号が＋か－かのようなことで動いているのです．このビットを8ケタ並べたものを1バイト（1 byte）と呼びます．1ビットでは0か1しか表せないとすると，1バイトは00000000～11111111，2^8つまり，256種類の情報を表すことができるわけです．

MSXは8ビットコンピュータですから，一度に扱えるデータは1バイトです．すべての文字やコントロールコードに0～255の値が割付けてあるのもこのためで，メモリの内容も1バイト単位で読み書きされます（「コンピュータの正体」参照）．

こういうわけですから，グラフィックス（たとえばSPRITE$）などを扱っていても，ビットやバイトはしばしば登場します．

次のようなSPRITE$を定義して画面に表示したければ，そのデータは8バイトになることがわかります．

2進数	10進数
10000001	129
10011001	153
10100101	165
11000011	195
10100101	165
10011001	153
10000001	129
00000000	0

20．ビット/バイト

2進数

ここで，2進数についてもう少し復習しておきましょう．10進数は0から9の数字を使い，

0，1，2，3……9，10，11

のように数えます．9の上はもう数字がないのでケタをあげて10になり，99の次は100という具合です．1983は

1	9	8	3		→	1	9	8	3	
千	百	十	一	の位		10^3	10^2	10^1	10^0	の位

と考えられます．

一方，2進数は0と1だけですから，

0，1，10，11，100……

と数えます．すぐにケタが上がるので親しみにくいという人もいるかもしれませんが，次のように考えれば何でもありません．たとえば10101は，

1	0	1	0	1		→	1	0	1	0	1	
16	8	4	2	1	の位		2^4	2^3	2^2	2^1	2^0	の位

1バイトは8ケタですから，最大は

1 1 1 1 1 1 1 1 ＝255
2^7 ………………………… 2^0

0～255が表せます．(11111111に1を足すと2^8……256)

LSB

一番右は2^0，つまり1の位で第0ビットとかLSB(Least Significant Bit)と呼ばれます．また一番左は2^7，つまり128の位で，第7ビットとかMSB (Most Significant Bit) と呼ばれます．

MSB

それにしても，2進数を10進数に変換するのはめんどうですし，2進数のままではケタが多くて扱いにくく，見にくいものです．そこで，2進数との変換が楽で，ケタも少ない16進数というものが考えられました．2進で2種類，10進で10種類ですから，16進では16種類の数字を使います．

16進数

0，1，2……8，9，A，B，C，D，E，F

20. ビット/バイト

10はA，11はBという具合で，0～15までは1ケタで表されます．16以上になるとケタが上がり，

10，11……1A，1B……1F，20……9F，A0……FF，100

という具合になりまます．たとえば3Fなら，

$$\underset{16^1}{3}\ \underset{16^0}{F} = 3\times16+15\times1=63$$

ということになります．3ケタ目は16^2（256）の位，4ケタ目は16^3（4096）の位になることは想像できると思います．ところで，16は2^4ですから2進数の4ケタ分を1ケタの16進数で表すことができます．

2進		16進		10進
0000	=	0	=	0
0001	=	1	=	1
0010	=	2	=	2
⋮		⋮		⋮
1001	=	9	=	9
1010	=	A	=	10
1011	=	B	=	11
1100	=	C	=	12
1101	=	D	=	13
1110	=	E	=	14
1111	=	F	=	15
10000	=	10	=	16

この性質を利用すれば，2進数を16進数に変換するのは簡単です．

$$\underbrace{1101}_{D}\underbrace{1010}_{A}\qquad\underbrace{1111}_{F}\underbrace{0011}_{3}$$

20．ビット/バイト

このように，1バイト（8ビット）の2進数は，2ケタの16進数で表現できるわけです．

8進数

この他，MSX BASICでは0～7の数字だけを使う8進数も用意されています．

1　2　=10
8^1　8^0 の位

これを使うと3ビットの2進数は1ケタの8進数で表せます．

001011101
1　3　5

これらの数が実際のMSX BASIC上でどう表現されているか，175という値を例にとってまとめておきます．

	2進	8進	10進	16進
表　現	&B10101111	&O257	175	&HAF
文字列を求めるとき	BIN$(175)	OCT$(175)	STR$(175)	HEX$(175)

10進以外の表現で，&B，&O，&H以下の部分がその数値になりますが，これをBIN$，OCT$，HEX$によって求めたときには文字列として与えられます．つまり，

文字列 P28

```
N=&HAF                    }
N=VAL（"&H"+ "AF"）       } N=175
N$=HEX$（175）              N$= "AF"
```

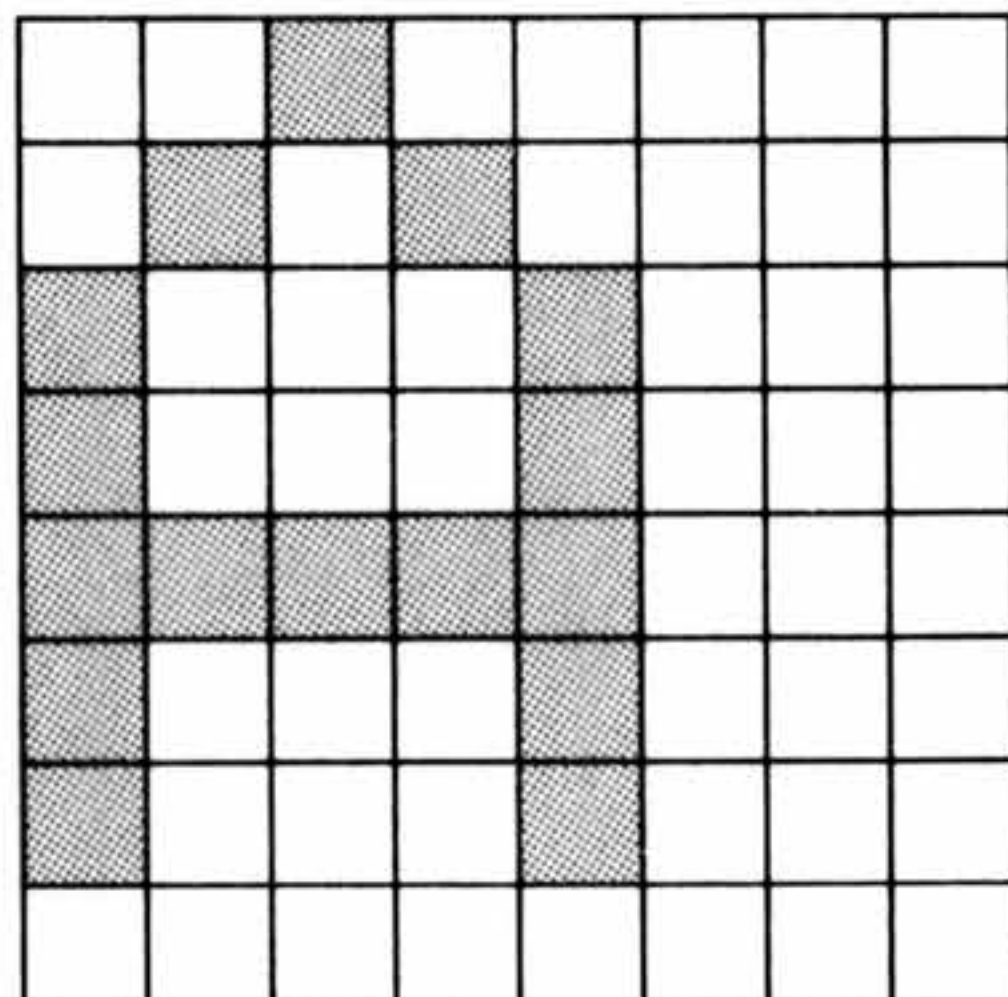
などは可能ですが

```
N$=&HAF                     &HAFは数値
N=HEX$（175）               HEX$（175）は文字列
N=AF                        AFは変数と解釈される
```

などは実行できない，もしくは正しく解釈されません．

21. フォント

フォントとは画面に出力される文字，キャラクタの構成のことで，１文字は８×８のドットでできています．たとえばAの大文字は，図のようなフォントになっています．本機に電源が投入されると，初期設定としてこれら（CHR$（0）～CHR$（255））の文字のフォントデータが，専用のRAM上にセットされます．１文字のデータは，SPRITEと同じように８ドットを１バイトとして８バイトですから，256文字文では2048バイトのデータが必要になります．

初期設定

このデータを書き換えてやれば，自分の思う通りの文字を定義して画面に出力することもできます．このデータが入っているRAMは，通常のRAMとは別にあるので，これに何かを書き込むときはVPOKE，読むときにはVPEEKを使います．

RAM
P 104

まずは，Aの大文字のデータを見てみましょう．

キャラクタコード

フォントデータは，そのキャラクタコード順に０番地から８バイトずつ並んでいます．CHR$（0）なら０～７，CHR$（2）なら16～23という具合です．Aのキャラクタコードは65ですから，65＊８＝520が先頭で，527が最後になります．次のプログラムを実行してみて下さい．

番地
P 102

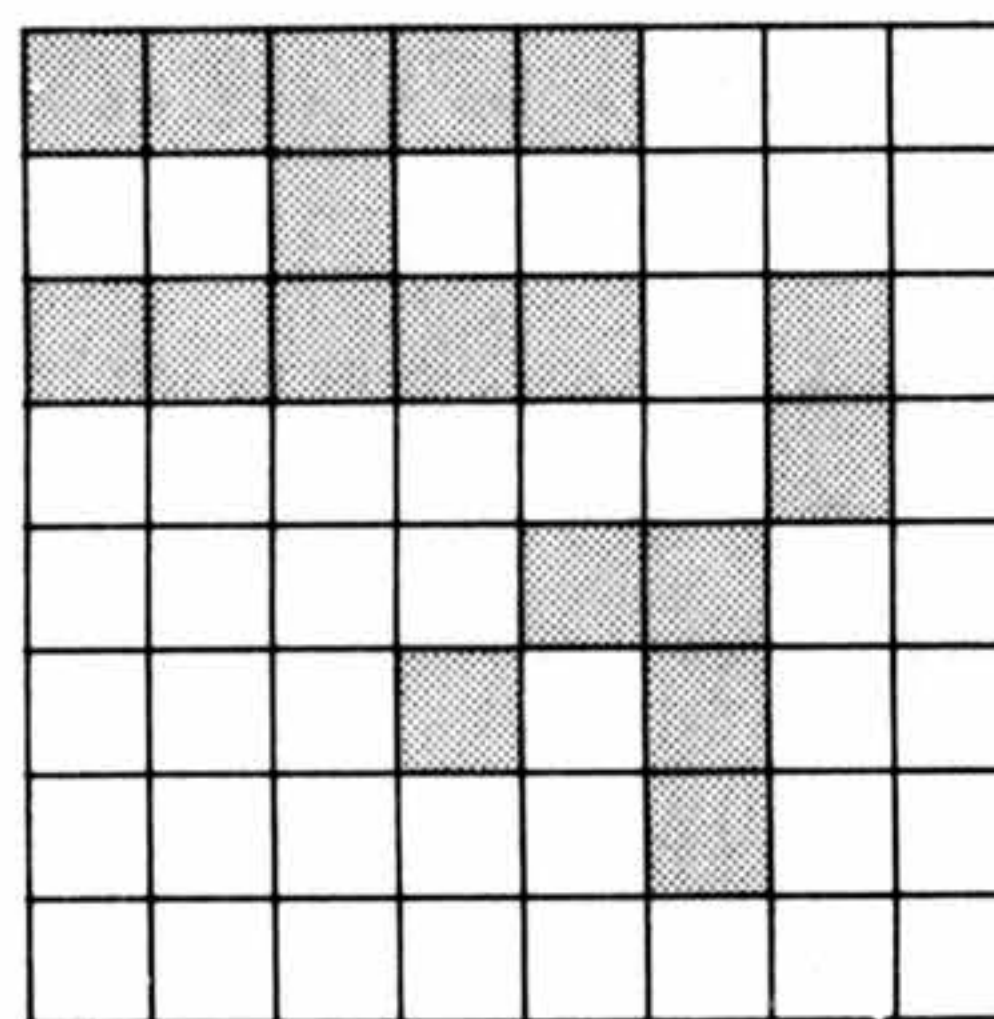
```
10 FOR I=520 TO 527
20 D=VPEEK (I)
30 PRINT RIGHT$ ("00000000"+BIN$ (D), 8)
40 NEXT
RUN
00100000
01010000
10001000
10001000
11111000
10001000
10001000
00000000
Ok
```

21. フォント

このように，SPRITE$のデータとまったく同じような形式でデータがあります．そこで，Aの大文字キー（またはCHR$ (65)）に次のようなフォントを定義してみましょう．

```
10  FOR I=520 TO 527
20  READ B$
30  VPOKE I, VAL ("&B"+B$)
40  NEXT
50  END
100 DATA 11111000
110 DATA 00100000
120 DATA 11111010
130 DATA 00000010
140 DATA 00001100
150 DATA 00010100
160 DATA 00000100
170 DATA 00000000
```

このプログラムを実行すると，画面に現れるはずのAの代りにエイという文字が表示されます．

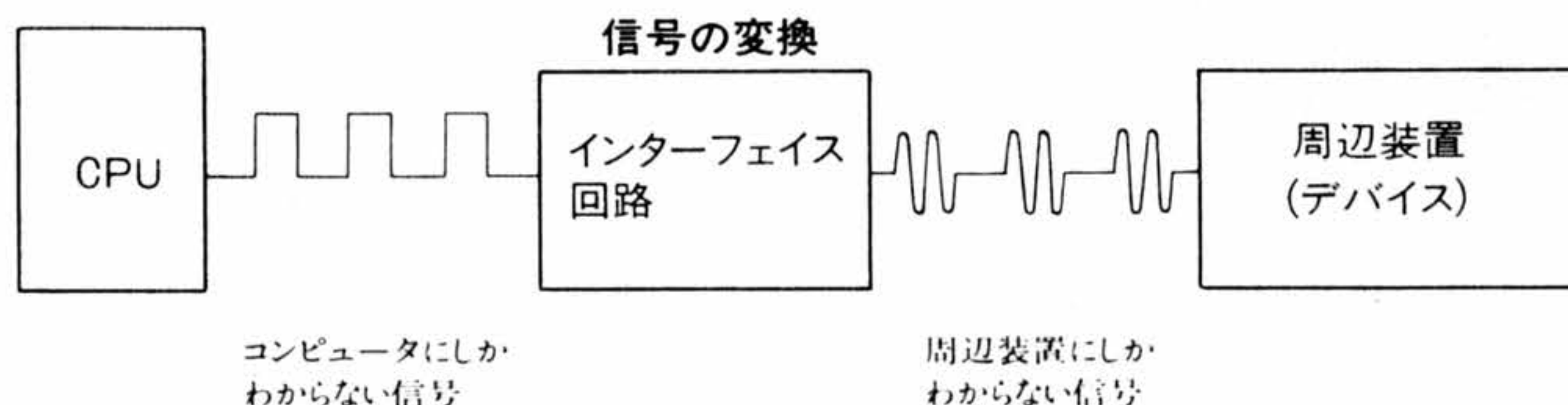
A～Z，a～zなどをあまり変えてしまうと，プログラムリストそのものまで見にくくなってしまいますから，実際に使うときは，CHR$ (123) 以降のかな文字や記号に何かを割り当てた方がよいでしょう．

ここまでの説明は，すべて画面モード1での場合を例にとっています．一度SCREEN 文を実行して他のモードにしたりすると，このフォントデータは初期化されることに注意してください．また画面モード0は，より多くの文字を表示するためにキャラクタは横6ドット×縦8ドットの構成になっています．各データの第0ビットと第1ビットは無視されると考えてください．また，このモードではデータの先頭（CHR$ (0) の最初のデータ）が2048番地となります．

初期化

22. インターフェイス

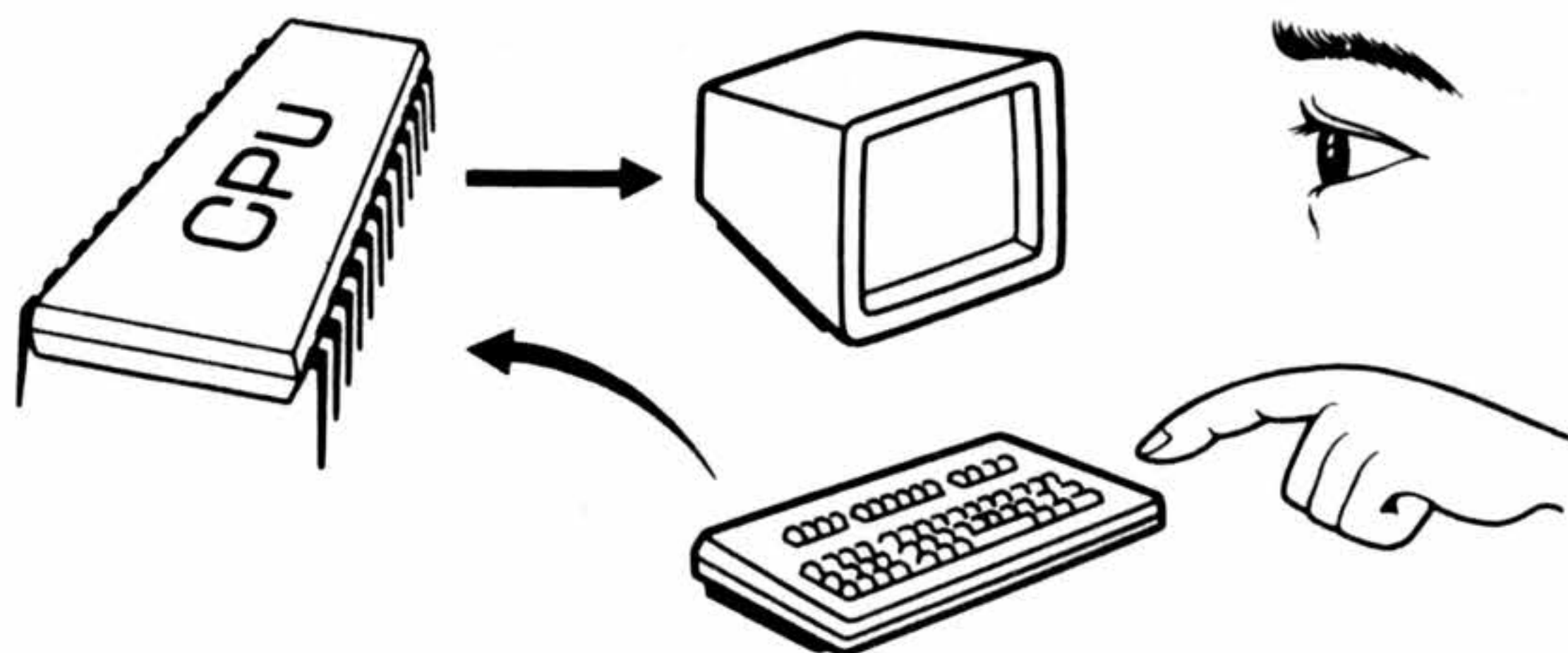
CPU P1

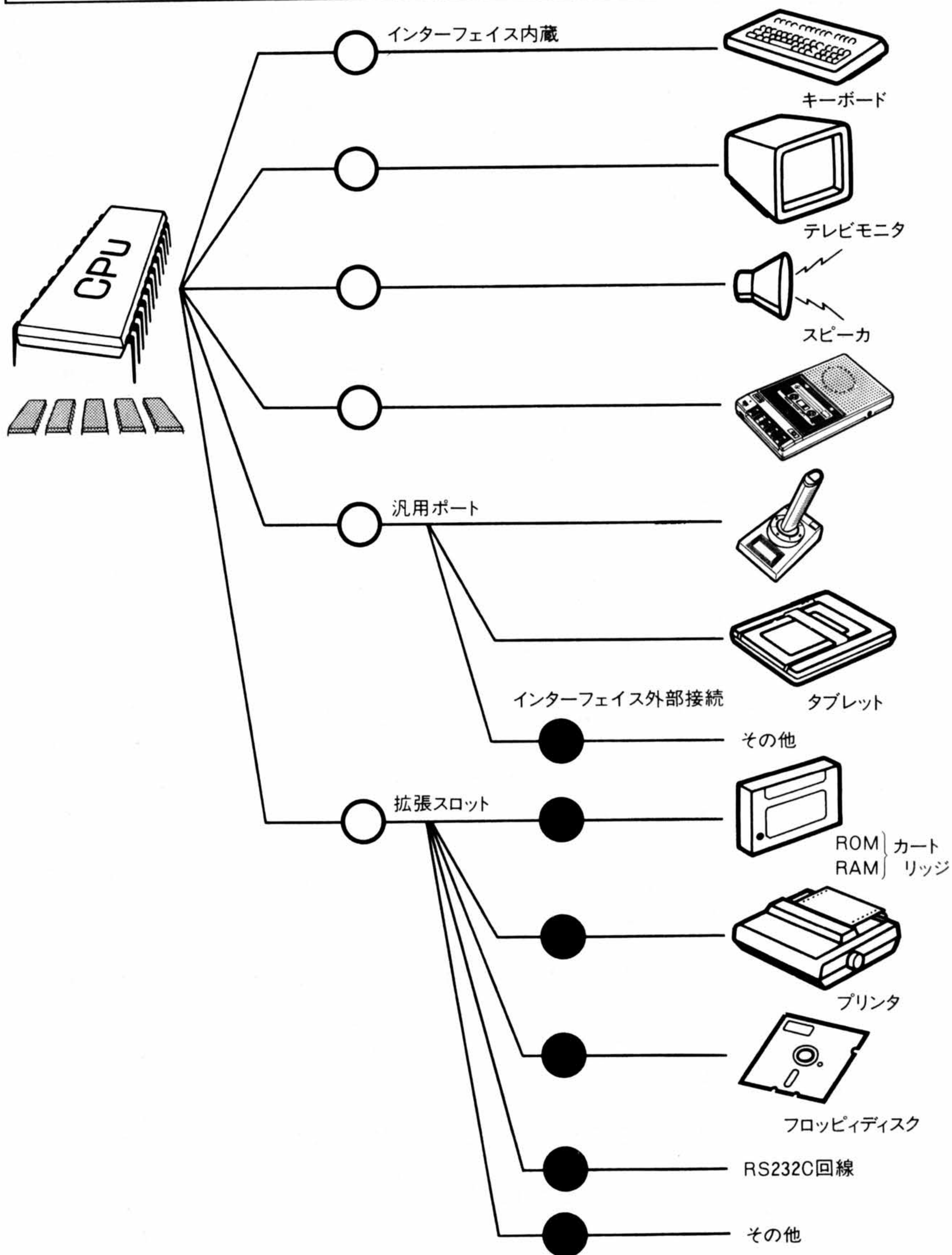
「コンピュータの正体」の項で，CPU が計算の他にさまざまな電気信号を決まった足から入出力する様子が解説してあります．ところが，その対象はさまざまで，信号の意味も違えばその伝達のスピードも物によって違います．相手がカセットテープなら，信号を音の信号にしてからひとつひとつゆっくりと（テープの回転が遅いので）カセットレコーダに送りますが，ディスクが対象であれば音に直す必要もなく，信号を比較的速く連続して送ってやることができます．このように，相手との信号のやりとりに必要になるのがインターフェイスです．

ディスク P13

デバイス

インターフェイスとは Interface と綴り，「顔と顔とのつき合わせ」というような意味で，CPU と周辺装置（デバイス）との仲介役と考えていいでしょう．現実には，いくつかの IC（集積回路）などからなる回路のことで，MSX ではキーボード，カセットレコーダ，テレビモニタ，ジョイステックやタブレット用のインターフェイスが標準実装されているので，これらの機器は自由に扱うことができます．更に，拡張スロットにインターフェイス回路をつなげばプリンタ，フロッピーディスク，RS232C 回路なども扱うことができます．

拡張スロット P (24)
RS232C 回線

また，インターフェイスという言葉は，もっと広い意味に使われることもあります．たとえば，CPU（コンピュータ）と人間の仲介役というように解釈すれば，テレビモニタやキーボードそのものがインターフェイスであるともいえるわけです．

22．インターフェイス

インターフェイス内蔵
キーボード
CPU
テレビモニタ
スピーカ
汎用ポート
タブレット
インターフェイス外部接続
その他
拡張スロット
ROM
RAM
カートリッジ
プリンタ
フロッピィディスク
RS232C回線
その他

23. ファイル

「ファイル」というコンピュータ用語は，初めのうちはなかなか理解できないかもしれません．その原因は，ファイルというものがバインダでとじた紙の束であるという先入観を捨てきれないところにあるのではないでしょうか．

実は，「ファイル」というのは意味を持った情報の集まりのことで，本でも，レコード，テープでもその中に文字や記号，音などがはいっているファイルということができるのです．

さて人間ならば，数値データなどをまとめて書いてとじておくにはバインダでもよいのですが，コンピュータでは紙の束をめくりながら文字を読むということは不可能です．そこで，このバインダの役目をするものは，すべてファイルと呼ぶことにしたのです．コンピュータでは情報の入出力は電気信号で行われるので，これらをファイルとしてとっておくにはカセットテープやディスクが最も便利です．

こうして作られるファイルには，大きく分けて2つの種類があります．

プログラムファイル

1つはプログラムファイルです．プログラムはコンピュータにとって最も重要な情報ですから，これには特別なコマンドが用意されています．

CLOAD, CSAVE, LOAD, SAVE, MERGE

これらのコマンドは，外部（主にカセットテープ）のファイルをメモリのテキストエリアにコピーしたり，逆にそのテキストを外部にコピーします．

データファイル

もう1つはデータファイルで，たとえば，電話番号帳の名前と番号などを順番に蓄えるためのものです．これを扱うには

OPEN, PRINT #, INPUT #, LINE INPUT #, CLOSE, MAXFILES

というコマンドを使います．

手順としては

OPEN "CAS : TELE" FOR OUTPUT AS #1

のようにテープ上に "TELE" というファイルを作り，

PRINT #1, "名前"

23．ファイル

のようにデータを書き込み，最後には

```
CLOSE
```

でファイル作りを終了します．

これを読むときは

```
OPEN "CAS : TELE" FOR INPUT AS #1
```

としてから

```
INPUT #1, A$
```

のようにします．

すべて読み終わったら

```
CLOSE
```

としておきます．

OPEN 文の構造は

```
OPEN "CAS:TEL" FOR OUTPUT AS #1
     └デバイス┘└ファイル名┘  └──モード──┘  ファイル番号
      (装置)
     │                │
     └ファイルディスクリプタ┘
```

となっていますが，MSX のファイル入出力はカセットテープが基本なので"CAS:" というデバイス名は，常に省略することができます．

デバイス

ファイル番号の方は，通常1だけです．というのも，カセットテープでは一度に読み書きできるファイルは1つだけだからです．

OPEN 文は，ファイルを読み書きするために用意するのですが，中には，書くこと（出力）しかできないファイルもあります．

```
OPEN "GRP : " FOR OUTPUT AS #1
OPEN "CRT : " FOR OUTPUT AS #2
OPEN "LPT : " FOR OUTPUT AS #3
```

などがそれです．

23．ファイル

これらは，モニタ画面そのものや，プリンタをファイルの装置(デバイス)としているので読み込む (INPUT：入力) ことができないのです．また，これらのデバイスで用意できるファイルは1つだけで，あとで読み出すこともないので〝ファイル名〃の指定はできません．

また，これらのファイルを例のように#1，#2，#3としておくと，PRINT #1ならばSCREEN2へ，#3ならプリンタへ書くということができます．このように1以外のファイル番号を使いたいときは，MAXFILES文で

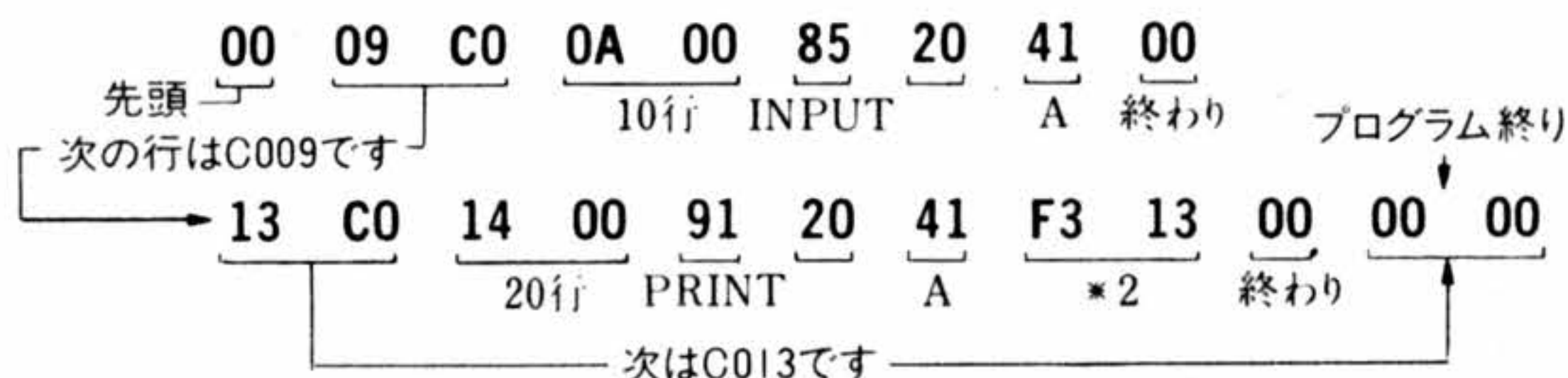
```
MAXFILES=3
```

ファイル番号　のように，最大のファイル番号を定義しておかなければなりません．

24．アスキー(ASCII)形式とバイナリ形式

BASICのプログラムがメモリの中で実際にどんな形になっているのかは,「プログラムの作成と実行」の項で詳しく解説してあります．たとえば

```
10 INPUT A
20 PRINT A * 2
```

というプログラムは，16進で書くなら&HC000～&HC014番地に次のような形で格納されます．

16進数 P 89

番地 P 102

これはBASICの実行を速くし，テキストを少ないメモリに収めるためにこうしておくもので,「バイナリ形式のプログラム」と呼びます．

テキスト

BASICインタープリタがBASICを実行するためには，必ずこの形でなければなりませんが，人間がキーボードから打ち込んだり，LISTコマンドでプログラムを見るときには，これでは困ります．そこで[1]，[0]，[I]，[N]，[P]，[U]，[T]……のように，プログラムが単なる文字の集まりとして入出力，あるいは記憶されたりするとき，これを「アスキー形式のプログラム」と呼びます．

インタープリタ P 3

人間が使うときはアスキー形式，コンピュータが実行するときはバイナリ形式になることはこれで明らかです．LISTというコマンドを実行すると，コンピュータはテキストエリアのプログラムをいちいちアスキー形式に直して画面に出力してくれるのです．

ところで，このプログラムをどこか外部に(カセットテープやディスクに)記録しておこうというときはどちらがよいのでしょう．

バイナリ形式で記録する場合であれば，SAVEするときは，結局メモリの中のものを順に出力して，カセットテープなりディスクなりに書き込むだけで済みます．読み込むときもそのデータを順に読んで，メモリに書き込むだけです．それに対して，アスキー形式ではLISTと同じように一行ずつ文字の並びに直してから，その文字を一文字ずつアスキーコード（キャラクタコードと同義と考えてください．Aなら65，bなら98のように決まっています）にしてカセットテープなどにSAVEし，LOADのときは，その逆を行いま

キャラクタコード

す．つまり，キーボードから一字ずつ打ち込んで RETURN とするように，テープから一字ずつ読み込んで RETURN があったら，それを一行としたバイナリ形式にしてメモリに格納するということになります．

このように，バイナリの方がSAVE，LOADも速くでき，データ量も少なくて済むので，一般にはプログラムはバイナリ形式で記録されます．

しかし，手間のかかるアスキー形式のプログラムにも大きな利点があります．たとえば，2つのプログラムを融合するようなときです．今，テキスト上に，

```
10 INPUT A
20 PRINT A * 2
```

というプログラムがあるとき，

```
5 REM CALC
30 END
```

の2つの行を加えるには，それらを単にキーボードから打ち込めば済みます．

エディタ

BASICのエディタは強力なのでこの5行を入力したときに，テキストの内容をみてちゃんと10行の前に挿入してくれます．つまり，1行入力するごとにテキストの内容は，そのすべてが完全になるように毎回書き換えられるのです．

この「キーボードから打ち込む」という作業のかわりに，アスキー形式のプログラムファイルを使うことができます．具体的にはMERGEコマンドを使うのですが，要するに，MERGEによってカセットテープ上（またはディスクなど）のアスキーコードが一文字ずつ読み込まれ，RETURN があるとそれを1行としてテキストに加えるという作業が行われるわけです．もしこのファイルがバイナリ形式だったら，単にその中の内容を1バイトずつ読んでメモリに書き込むだけですから，MERGEは不可能です．これは，つけ加えるプログラムが大きいときなどは非常に便利です．

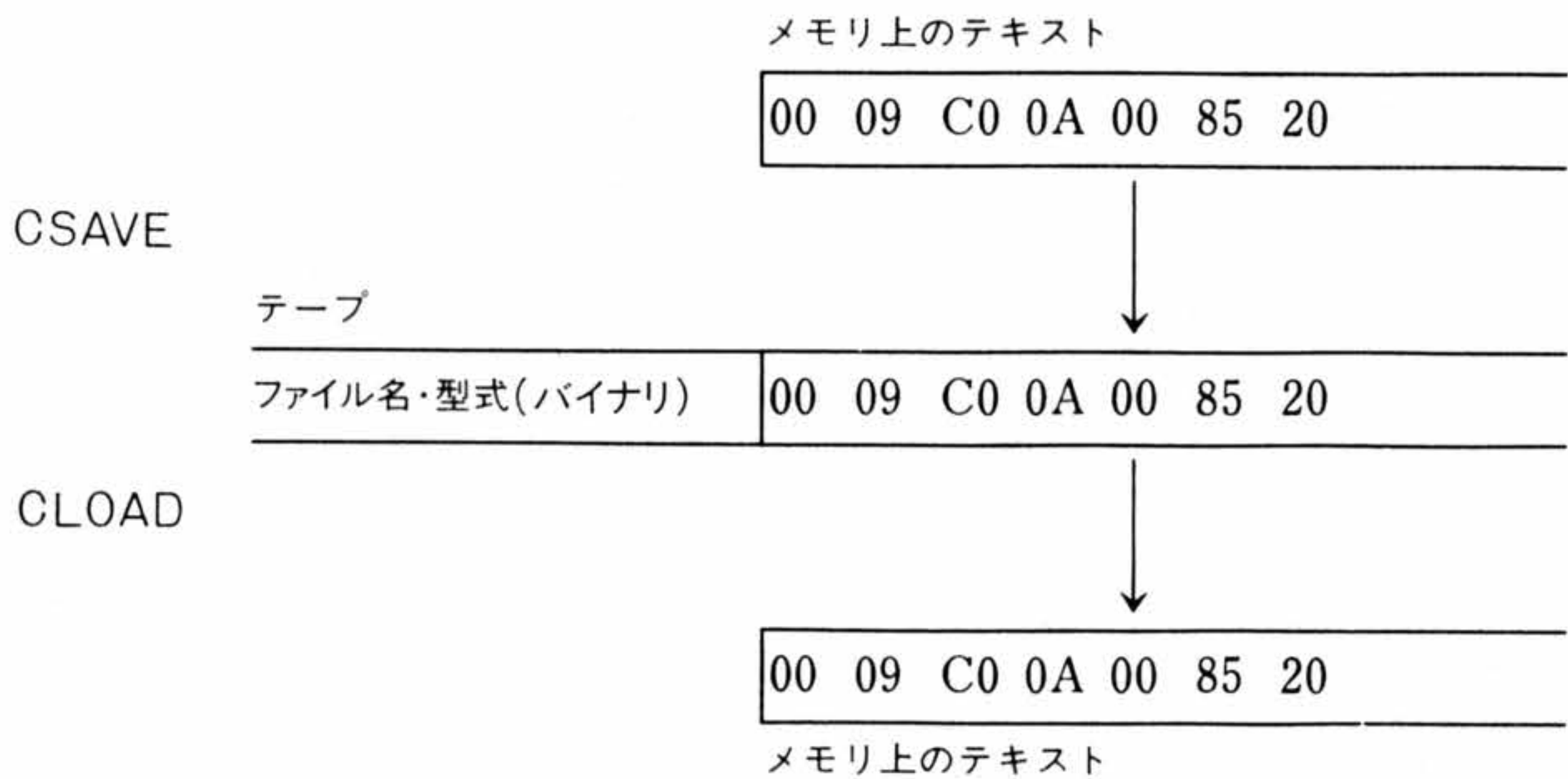
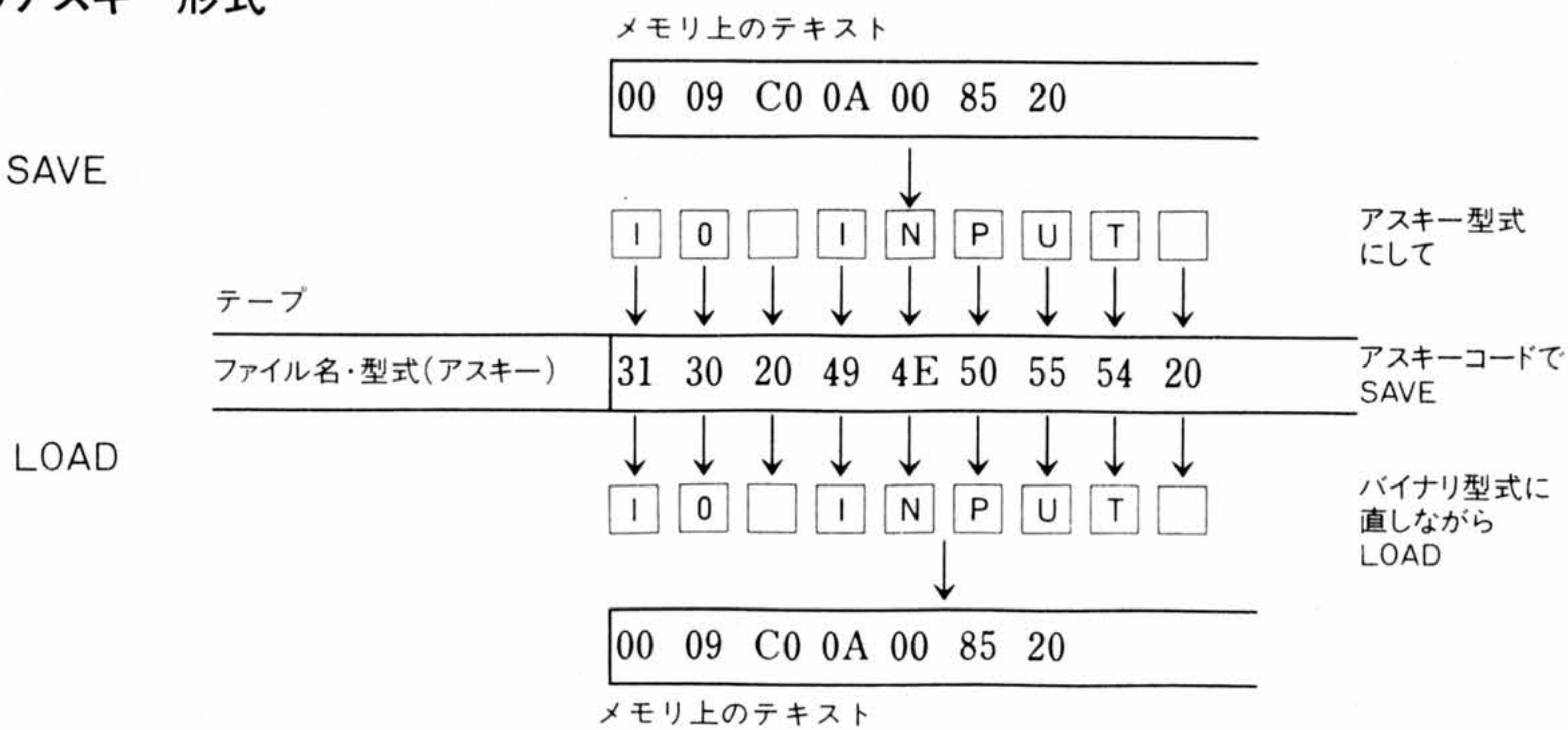
もう1つの利点は，たとえば，MSX以外のBASICのテキストでもアスキー形式ならLOADできることが多く，LOADしたあとに，少し修正を加えれば正しいプログラムにできることです．いろいろなBASICもLISTした状

24. アスキー(ASCII)形式とバイナリ形式

態では，あまり違いはありませんが，バイナリだと中間コード(INPUT は133などの決め方）や行の構造が違っていることが多く，そのままメモリにロードしてもまったく機能せずに，修正も不可能なことさえあるのです．

中間コード

次に示すのは，プログラムをカセットテープにSAVE, LOADする様子です．

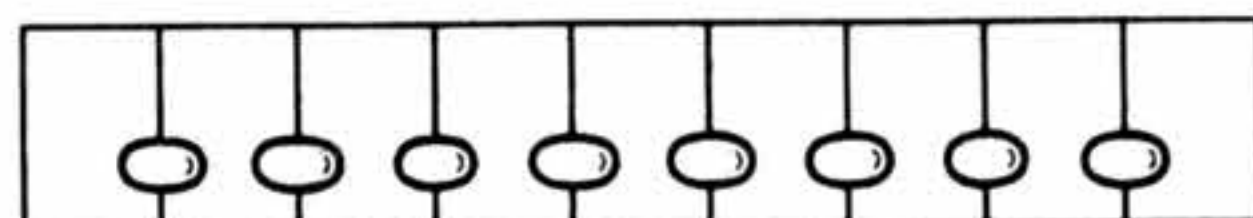
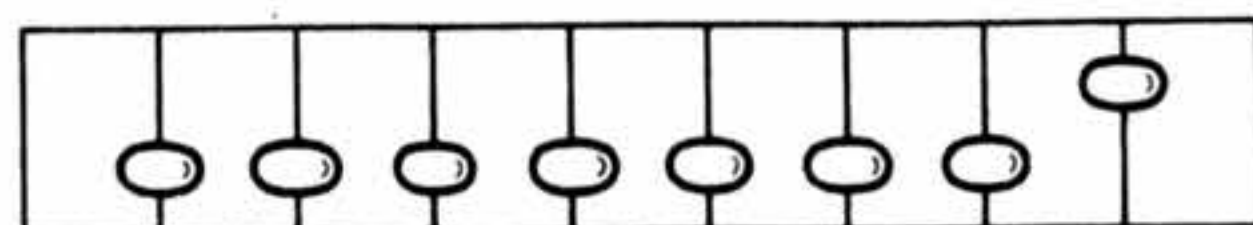
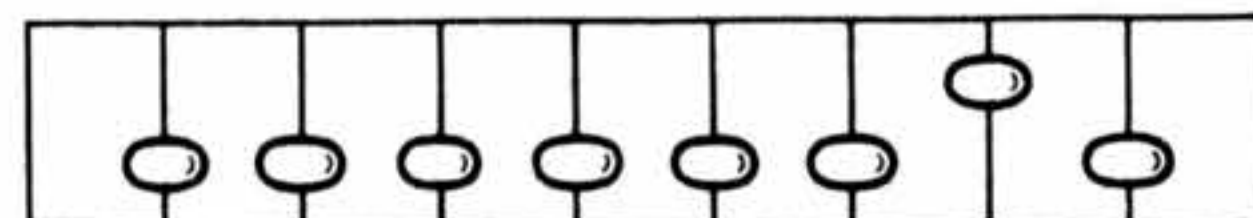
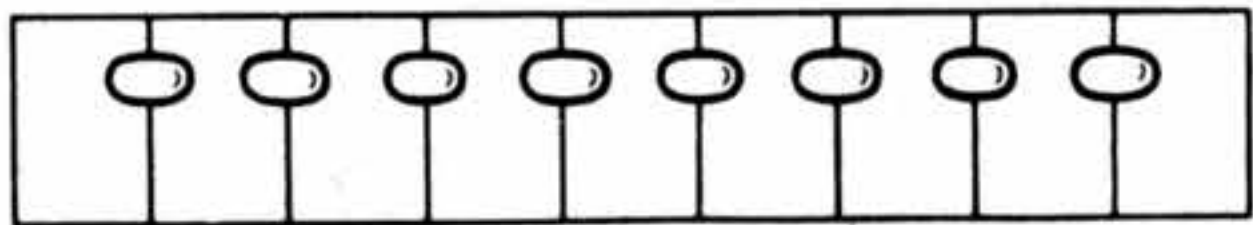
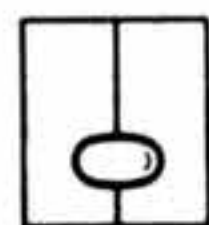
プログラムをテープにSAVE, LOADする様子

アスキー(ASCII)とはAmerican Standard Code for Information Interchangeの略で各文字をコンピュータ間で転送するときなどのために割り当てられた文字のコードです。

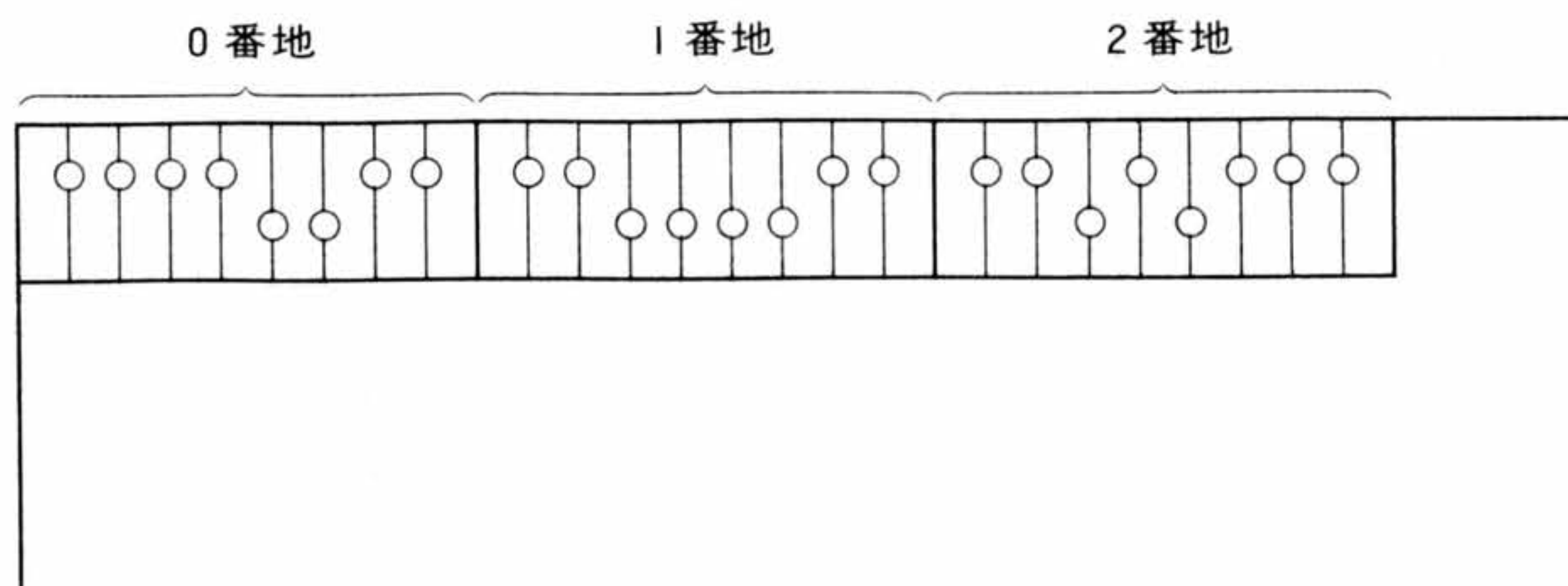
25．メモリ

「コンピュータの正体」の項を読めばわかると思いますが，コンピュータの主役はなんといってもすべての入出力と計算を司るCPUです．しかしCPUは記憶能力はほとんどなく，Z80ではわずか26組の8ビットデータを一時的に記憶できるだけです．それも計算用に使えるものは少ないので大きな数値を扱うとなったらもうお手上げです．そもそも，その計算手順(プログラム)はどこに記憶するのでしょう．

レジスタ

CPU内でデータを記憶する部分をレジスタと呼び，Aレジスタ，Bレジスタなどの名前を付けてあります．8ビットレジスタは次のようなものと考えてください．

番地

電子のそろばん玉．現在0です．
2進数で00000000

1を加えよとCPUに命令があったのでそろばん玉が1つ上に上がる．2進数で00000001

更に1を加えよと命令されると1＋1は2なので2進数で00000010

次に3を引けと命令されると，引けないのでとなりに上位のそろばんがあったものとして計算し，1玉借り2進数で11111111

Z80はこれを内部に26個持っていると考えてよいのですが，メモリもこれとほとんど同じものを一単位としています(計算はできませんが)．そして各単位には0から順に番号を付け，何々番地と呼んで参照します．0番地をのぞいて見ると243，1番地は195という具合に値が入っているわけですが，CPUがこの内容を見るためにはアドレスバスと呼ばれる電線でCPUとメモリがつながっていなければなりません．Z80は16本のアドレスバスがあるので2^{16}，つまり0～65535番地まで(16進数でFFFF)のメモリを見ることができます．

25．メモリ

● ROM と RAM

これでメモリとはどんなものかわかりましたが，メモリの内容はいったい何を表し，どうやって書いたものなのでしょう．それは「コンピュータの正体」の項に書いたとおりですが，CPU は電源がはいるとまず，0 番地から順にメモリ内容を見るようにできています．MSX BASIC では PEEK 関数がありますから，これを使って 0～3 番地の内容を見てみましょう．
？HEX$（PEEK（0））のように16進で出力させてください．

番地	0	1	2	3
	F 3	C 3	D7	02

CPU はこれらの内容を次のように解釈実行します．

F 3……割込み（詳しくは専門書参照）を禁止する
　　次は 1 番地を見る．

C 3……以下 2 バイトの内容(D7 と 02)を見て，それらを合わせた番地(02D7 番地）の内容を次の命令とする．
　　次は 02D7 番地を見る

このようにしてその 02D7 番地以降も次々と命令やデータが入っています．これらの命令を機械語と呼びますが，それは Z80 の解説書で勉強しなければなりません．

機械語

MSX は電源を入れるとすぐにこれらのプログラムが実行され，“Ok”と画面に表示されて BASIC 言語を受け付ける状態になったことを知らせてくれます．ところで，もしこの F3, C3 など最初の方の機械語プログラムが変化してしまったら大変です．BASIC を受け付けないとか，キーボードを受け付けないとか，とにかくコンピュータがただの箱になってしまうかもしれません．

そこでこのような大切な，あるいは変えてはならない部分には ROM（Read Only Memory）というメモリを使います．ROM の内容は工場で作られたときに決まり，その内容は二度と書き換えることができません．

ROM

試しに次のような実験をしてください．

25．メモリ

```
 ?PEEK (0)
  243
Ok
  POKE 0, 0
Ok
 ?PEEK (0)
  243
Ok
```

POKE文はメモリに内容を書き込むBASICの命令ですが，0番地(ROM)を相手にしては何もできませんでした．CPUがいくら頑張ってもROMの内容は変わらないのです．MSXでは0番地から&H 7FFF番地までがROMになっています．BASICを実行するためのプログラム（BASICインタープリタ）はそれほど大きなものなのです．

インタープリタ

さて一方，無事にBASICが使える状態になったら今度はそのBASICで書かれたプログラムや変数の値を覚えておくためのメモリが必要です．BASICプログラムはそれを作っている人が，次々と変えていくものですし，変数の値はプログラムの実行中に次々と変化します．そこで読んだり書いたりできるのがRAM（Random Access Memory）と呼ばれるメモリで標準のMSXでは&HC000～&HFFFF番地があてられています．&H8000～&HBFFF番地は別のことに使うので，最初はメモリがないものと考えてください．?PEEK（&H8000）などとしても毎回違った値が返ってきます．また&HF380～&HFFFFもCPUがBASICを正しく実行するための作業領域（ワークエリア）ですから，不用意にそこに値をPOKEして使ったりするとBASICが実行できなくなってしまいます．

RAM

電源を切るとRAMの内容は，すべて消えてしまいます．ですからBASICのプログラムなどは電源を切る前にカセットテープにSAVEしておかなければなくなってしまいます．このカセットテープのようなものをメモリに対して，外部記憶装置と呼ぶこともあります．一方CPUが使うワークエリアも電源を切れば皆消えてしまいますが，こちらの方は再び電源を入れたときにROM内のプログラムによって，CPUが勝手に作り直しますから心配はいり

25. メモリ

ません.

最後にRAMの内容のテストをしてみましょう.

まず電源をONにしたらすぐに?PEEK(&HC000)を実行してみてください.

```
 ?PEEK (&HC000)
  0
Ok
```

次にPOKE文で何か書き込んでからもう一度PEEKします.

```
POKE &HC000, 15
Ok
 ?PEEK (&HC000)
  15
Ok
```

15が書き込まれています.

次に一度電源を切り,また入れてからPEEKしてみます.

```
 ?PEEK (&HC000)
  0
Ok
```

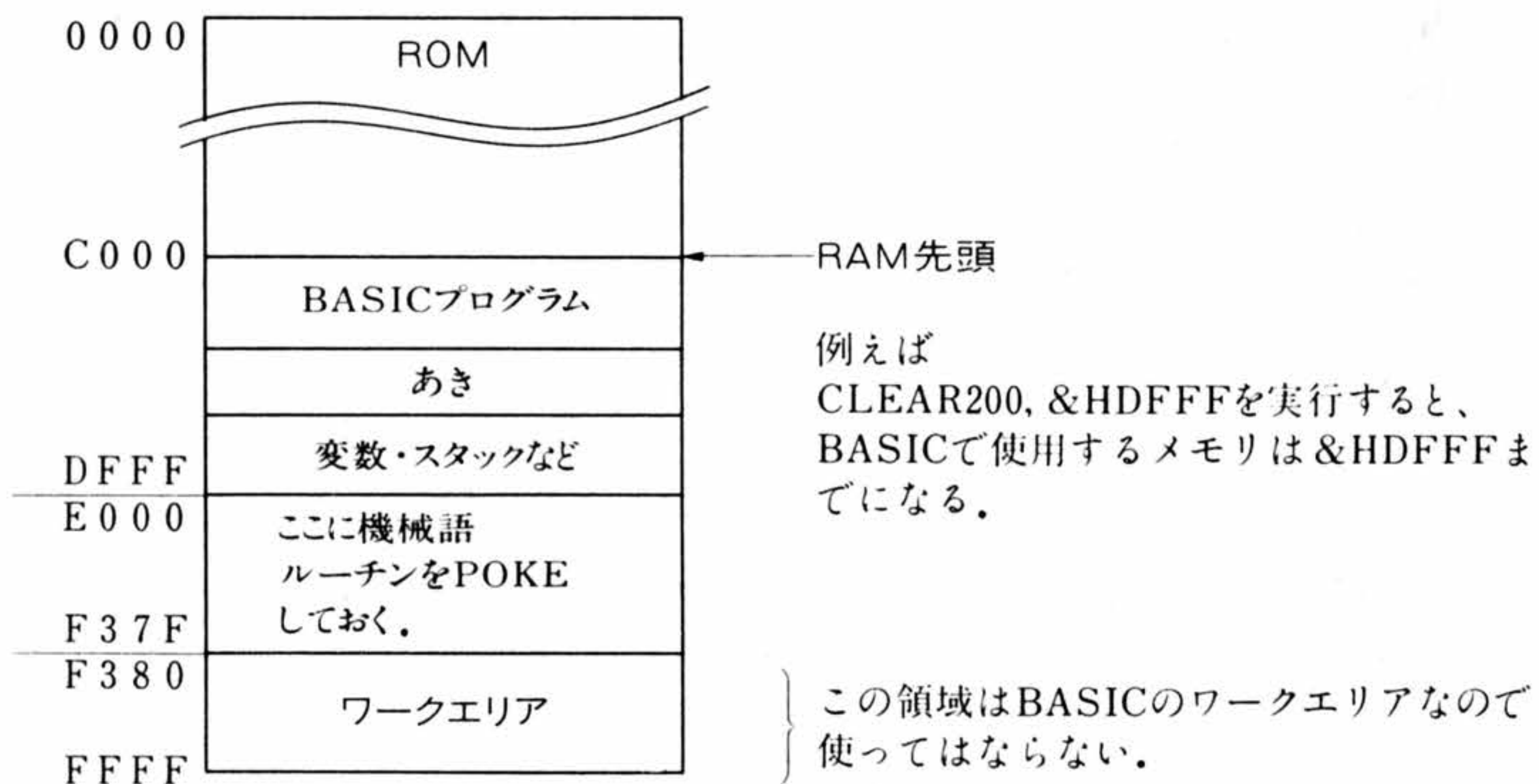
消えてしまいました.

26. 機械語

機械語とはどんなものか，知らない人はまず「コンピュータの正体」，「メモリ」，「ビット／バイト」などの項を読んでおいてください．また，機械語の命令そのものについては，Z80の機械語についての解説が載っている他の本で勉強してください．ここではBASICプログラム中から，USR関数を使って機械語サブルーチンを呼び出す場合について解説します．

サブルーチン

URS関数は，一般に次のような使い方をします．

① まず，プログラムの先頭でCLEAR文を実行し，機械語プログラム領域をRAM上に確保します．

② DEFUSR＝&HE000のようにしてルーチンの先頭を定義します．

③ POKE文を使って，確保した領域内に機械語プログラムを書き込みます．

④ 必要な箇所でA＝USR(B)の形でUSRルーチンを呼び出します．B(引数)の値はFACと呼ばれるワークエリアに移され，AおよびHLレジスタは，引数の型に応じて次のような値になります．

FAC

26. 機械語

引数が文字型の場合は，Aレジスタの値は3になり，ストリングディスクリプタの番地が値としてDEレジスタに入れられます．

ストリングディスクリプタ

次に，CPUの制御はDEFUSR文で宣言した番地に移り，機械語のRET命令(&HC9)によってBASICに復帰します．このとき，FACにUSR関数の結果を返しておけばその値が変数に代入されます．

A=USR(B)⟶ A=f(B)

この様子をごく簡単なサンプルで示します．0から254の値をもつ引数に1を加えて戻してみます．

```
10 CLEAR 200, &H DFFF
20 DEFINT A-Z
30 AD=&HE000: DEFUSR=AD
40 FOR I=0 TO 3
50 READ A$
60 POKE AD+I, VAL ("&H"+A$)
70 NEXT
```

26. 機械語

```
80 DATA 23,23,34,C9
90 'MAIN
100 INPUT "B=(0-254)" ; B
110 A=USR(B)
120 PRINT  "B+1=" ; A
130 END
```

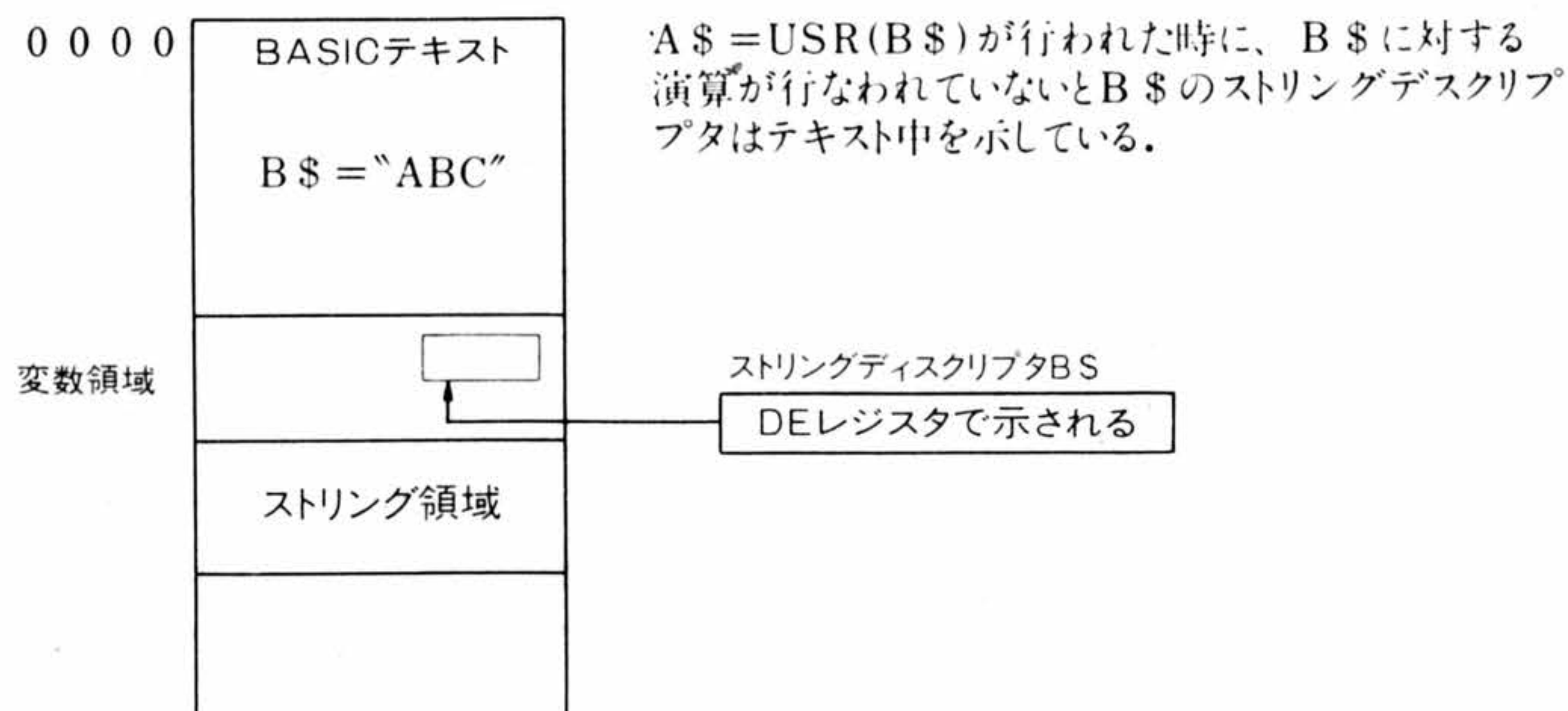
ソースリスト　**機械語部分のソースリスト**

ニーモニック

番地	コード	ニーモニック
E000	23	INC HL
E001	23	INC HL
E002	34	INC (HL)
E003	C9	RET

これは最も簡単な整数型の演算です．単精度と倍精度は FAC 上は同じ形で(HL)が指数，(HL+1)が仮数最上位 2 ケタを表しますが，これらは BCD 表現になっています(BCD 演算については機械語の専門書を参照してください)．

BCD

引数が文字型のときは注意が必要です．文字型変数は演算が行われていない場合，その実体はプログラムテキスト中にあり，ストリングディスクリプタもそこを示しています．

テキスト中の文字列を操作して長さが変わったりすると，BASIC のテキストが壊されてしまい，暴走の原因となります．これを防ぐには，A$=USR(B$+"")のようにします．これで B$は 1 度ストリング領域にコピーされ，

暴走

26．機械語

ストリングディスクリプタもそこをさします．

ストリング領域内でも，あまり不用意にその実体をいじれば，他の文字列変数（使っていればの話ですが）やストリング領域外を破壊しかねません．これを防ぐには，ストリング領域を十分大きく取った上で，引数の文字変数を十分な長さにしておくことです．

ストリングの長さを変化させるときは，このようにさまざまな注意が必要で，更には，ストリングディスクリプタの中の文字列の長さの値を書き換えておくことも忘れてはなりません．

FACを無視してしまえばUSR関数は，単に機械語ルーチンを呼び出すためのコマンドとして使えますが，形は関数ですので，A＝USR(0)のようにダミーの変数や引数を使ってください．また，複数個の引数や結果をやりとりするには，ユーザーが機械語領域内にワークエリアを決めて引数をPOKEしておきます．USR関数を呼び出した後に，その結果をPEEK関数で受け取ります．

最後に，実用例として，配列やVARPTR関数などを使った高速のソーティングルーチンを載せておきます．アルゴリズムは，なるべくわかり易いように単純挿入法を使っています．整数型の配列AにN個のデータを乱数によって代入し，USRルーチンで小さい順に並べ換えます．BASICで同じ作業をやると100倍もの時間がかかります．

```
10 CLEAR 200,&HF32F
20 DEFINT A-Z:WIDTH 30
30 AD=&HF330:DEFUSR=AD
40 FOR I=0 TO 73
50 READ A$:V=VAL("&H"+A$)
60 POKE AD+I,V
70 NEXT
80 DATA 23,23,5E,23,56,D5,DD,E1
90 DATA DD,5E,FE,DD,56,FF,1B,DD
100 DATA 4E,00,DD,46,01,DD,6E,02
110 DATA DD,66,03,DD,23,DD,23,DD
120 DATA E5,AF,E5,ED,42,E1,30,18
130 DATA DD,70,01,DD,71,00,DD,74
140 DATA FF,DD,75,FE,DD,2B,DD,2B
150 DATA DD,46,FF,DD,4E,FE,18,E1
160 DATA DD,E1,1B,BB,20,C9,BA,20,C6,C9
```

```
130 INPUT "No. of Data";N
140 DIM D(N):D(0)=0:I=RND(-TIME)
150 FOR I=1 TO N
160 D(I)=RND(1)*N
170 NEXT
180 PRINT "START !"
190 TIME=0
200 V=USR(VARPTR(D(0)))
210 V=TIME
220 FOR I=1 TO N
230 PRINT USING"#####";D(I);
240 NEXT:PRINT
250 PRINT V;"/60秒"
260 END
```

機械語部分のソースリスト

```
                          ORG  0F330H

F330  23                  INC   HL
F331  23                  INC   HL
F332  5E                  LD    E,(HL)
F333  23                  INC   HL
F334  56                  LD    D,(HL)      (DE)=D(0)の下位
F335  D5                  PUSH  DE
F336  DDE1                POP   IX          IX=DE
F338  DD5EFE              LD    E,(IX-2)
F33B  DD56FF              LD    D,(IX-1)
F33E  1B                  DEC   DE          DE=N(要素数-1)
F33F  DD4E00      STRT:   LD    C,(IX+0) } BC=D(I)
F342  DD4601              LD    B,(IX+1) }
F345  DD6E02              LD    L,(IX+2) } HL=D(I+1)
F348  DD6603              LD    H,(IX+3) }
F34B  DD23                INC   IX
F34D  DD23                INC   IX
F34F  DDE5                PUSH  IX
F351  AF          COMP:   XOR   A
F352  E5                  PUSH  HL
F353  ED42                SBC   HL,BC       比較
F355  E1                  POP   HL
F356  3018                JR    NC,OK       D(I)<=D(I+1)
                                            なので並べ替えなし.
```

26. 機械語

```
F358 DD7001           LD    (IX+1),B   ┐
F35B DD7100           LD    (IX+0),C   │ 並べ替え.
F35E DD74FF           LD    (IX-1),H   │ SWAP D(I), D(I+1)
F361 DD75FE           LD    (IX-2),L   ┘
F364 DD2B             DEC   IX
F366 DD2B             DEC   IX
F368 DD46FF           LD    B,(IX-1)
F36B DD4EFE           LD    C,(IX-2)
F36E 18E1             JR    COMP
F370 DDE1       OK:   POP   IX
F372 1B               DEC   DE          終了判定
F373 BB               CP    E
F374 20C9             JR    NZ,STRT
F376 BA               CP    D
F377 20C6             JR    NZ,STRT
F379 C9               RET
```

27. エラー

インター
プリタ

MSX システムは電源がはいるとすぐにその BASIC インタープリタ・プログラムが実行されるようにできています．この辺の事情は「コンピュータの正体」や「プログラムの作成と実行」に詳しく解説してありますが，MSX は電源を投入後は USR 関数を使わない限り常にインタープリタ・プログラムの管理下にあるといっていいでしょう．

インタープリタは BASIC のプログラムを実行しているときに実行不可能な命令を検出すると，対応するエラーメッセージを出力して実行を中断し，コマンドレベルに戻ります．これを BASIC のエラートラップ機能と呼びます．エラーというものは初めのうちは，たくさん出てきていやになる人もいるかもしれません．しかしコンピュータ側では BASIC の文法にない命令や 5÷0 のように不可能な演算に出会ったとき勝手に実行を進めることはできないので，「どうしましょう」と人間に対処を求めていると考えればいいでしょう．

エラートラップ

初めのうち，よく出してしまうエラーには Syntax error, Illegal function call, などがあります．プログラムをどう直していいかわからないときなどは「トラブルシューティング」の項を参照してください．

エラーの原因となるプログラムの間違いを「バグ」と呼び，これを見つけて直すことを「デバッグ」といいます．バグの中でも大変厄介なのは，エラーも出さず実行も中断しないが予想外の実行結果になるというようなものです．これも「トラブルシューティング」で解説しているので参考にしてください．

デバッグ

さて，もう一つのケースとしてエラーが出ることはわかっているがなんとか実行を止めずにプログラム中で処理してしまいたいことがあります．たとえば，ある数の平方根を出すプログラムでマイナスの引数のときは虚数として出力したいとします．こんなときには，ON ERROR GOTO 文が役に立ちます．

```
10 ON ERROR GOTO 100
20 INPUT "X="; X
30 PRINT SQR(X) ;
40 IF F=1 THEN PRINT "i"  ELSE PRINT
50 F=0 : GOTO 20
```

27. エラー

```
100 X=-X: F=1
110 RESUME
```

このとき１つ注意しなければならないのは，100～110行のエラー処理ルーチンではXを正に戻して，そのことを伝えるFを１とする以外何もせずに同じ箇所 SQR(X) を再トライしていることです．もし20行あたりが

```
20 INNPUT "X=";X
```

などと Syntax error になっていたら何度エラー処理をしてもこの Syntax error は解決されず20行と100，110行で無限ループができてしまいます．こういったミスを避けるには，100行以降を

無限ループ

```
100 IF ERR=5 AND ERL=30 THEN X=-X: F=1: RESUME
110 PRINT ERR; ERL
120 END
```

のようにして，予想外のエラーはきちんと検出できるようにするとよいでしょう．

また ON ERROR GOTO 文を含むプログラムの実行を中断してコマンドレベルに戻ったとき，この宣言はまだ有効なままです．このときにエラーを出すと，エラー処理ルーチンの実行が始まってしまいます．宣言を取り消すには，ON ERROR GOTO 0 や CLEAR を実行してください．また，プログラムを書き換えたり，NEW, RUN，を実行したときも宣言は無効になります．例のプログラムを単に RUN すると再び10行で宣言されますが RUN 20 のようにすると宣言は無効になります．

28．トラブルシューティング

本機を使っていて何か困ったことがでたときのために，その解決のヒントとなるような点をいろいろな場合を想定して掲げておきます．

1．起動時

まずは次の点に注意してください

- 各スイッチはONになっているか
- 各コネクタなどは正しくしっかりと差し込まれているか，入門編の「セットアップ」を参照する．
- CLOADがうまくいかないときは，入門編の「オーディオカセットレコーダ」を参照する．
- キーボードの使い方やプログラムの作り方については，入門編の「キーボード」を参照する．
- コンピュータで何ができ，何ができないかがわかりにくいときは，「コンピュータの正体」の項などを読み，BASIC説明書のサンプルプログラムを入力するなどして，とにかく使ってみること．

プログラム

コンピュータはプログラムを与えなければ何もできません．まずは他人が作ったプログラムを利用するのが手軽ですが，本当に使いこなすには，プログラミングを少しでも覚えてしまった方が有利です．MSXはプログラムによって壊れてしまうことは絶対にありません．たとえ，POKE文やUSR関数などを使って壊れてしまったように見えても，電源を切り，再び入れれば必ず正常になります．プログラムを作って実行させることを恐れることはありません．

また「こんなつまらないプログラムを作っても」などと考えるのは禁物です．誰もが初めから大きな有益なプログラムを作れるわけではありません．英会話の勉強のように「数をこなすこと」「習うより慣れること」が大切です．BASICの単語などは200にも満たず，文法も極めて単純です．英会話などよりは，ずっと簡単に使いこなせるようになるはずです．

28．トラブルシューティング

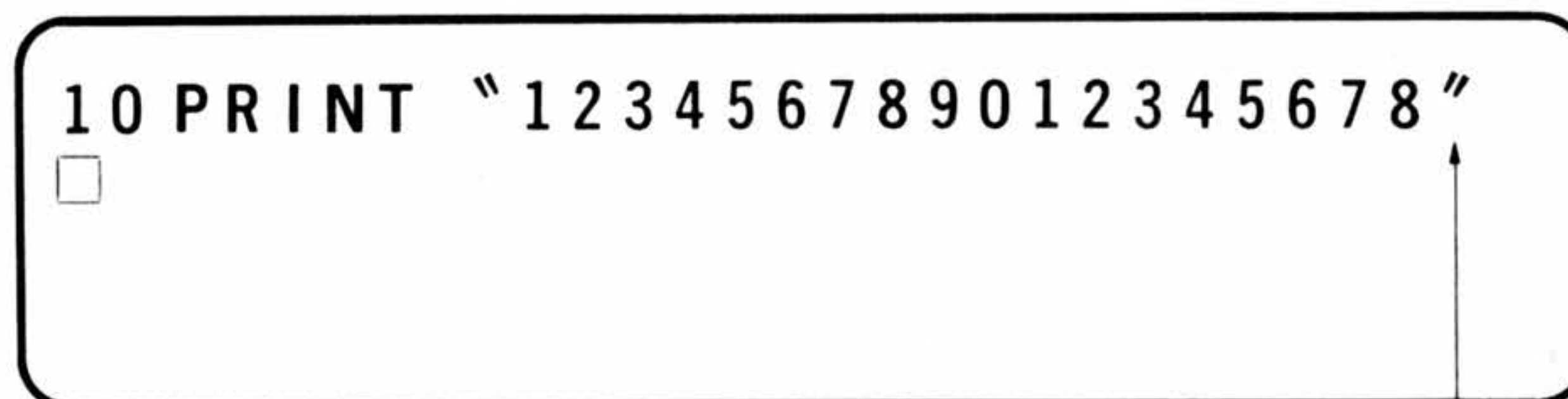
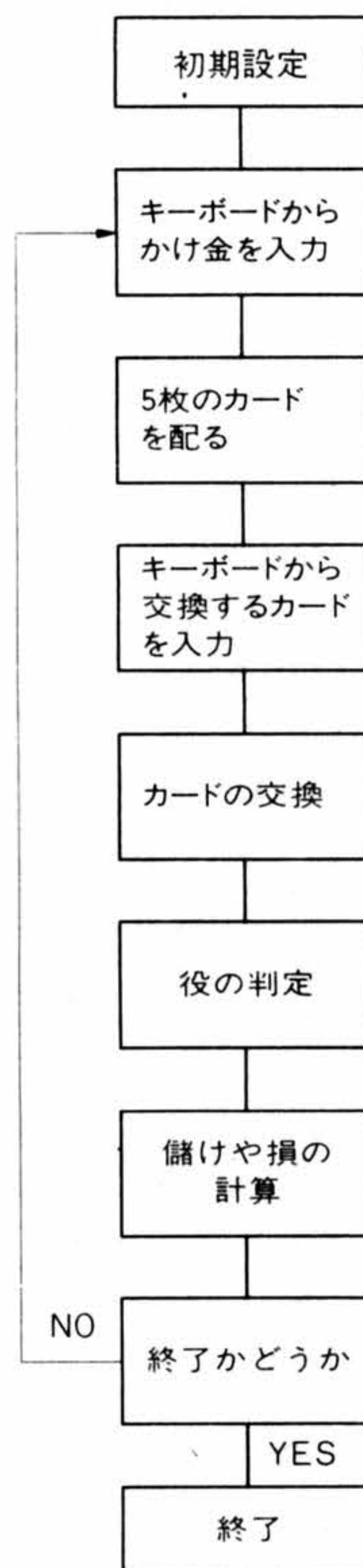
2．　プログラムを作るとき

小さなプログラムであれば，いきなりキーボードから入力し，実行しながら直してゆけばできてしまうかもしれません．しかし，ちょっと大きくて役に立つ，あるいはおもしろいプログラムを作るとなったら，フローチャートや変数表は必要不可欠となります．

フローチャート

まず，人間は何を入力し，コンピュータはそれをどう処理してその結果を返すか，どういう結果がでたらプログラムを終了するか，などをしっかりと決めておかなければなりません．

たとえば，１人ポーカーのゲームを作りたいなら，右のような大まかな流れ図（ジェネラルフローチャート）を書き，更にひとつひとつの処理を細分化してフローを書きます．このときどんな変数を何に使うか，見逃している点はないか（たとえば，持ち金以上にかけ金を入力したらどうするかなど），気を配りながらBASICのプログラム行にしていきます．画面表示の様子なども実際にどうなるのか，まずはPRINT文，LINE文などで絵だけを作ってみるのも大切なことです．

こうした作業をするときでも，日頃小さなプログラムを作り慣れていると，どこをどのようにプログラムするかが適確にわかるものです．

次に，実際にプログラムを入力するときの注意点をいくつか掲げておきます．

28．トラブルシューティング

● プログラムの１行をタイプし終ったら必ず RETURN キーを押す．次のような場合は，特に忘れやすいので注意する．

表示文字29のとき（WIDTH 29）

```
10PRINT "1234567890123456 78"
□
```

29文字目をタイプするとカーソルは次に進む

これで10行が終りなら，ここで必ず RETURN キーを押す．これを忘れて，20 PRINT……などと続けてしまうと 20 PRINT……は10行の一部になってしまい，正しいプログラムにならない．LIST したとき RETURN キーを押し忘れていなければ

```
10 PRINT "12345678901234 5678"

20 PRINT "ABC"
```

のように，10行と20行の間は１行あく．

テキストエリア

● 新しいプログラムを作るときは，まず LIST を実行して，テキストエリアに前のプログラムが残っていないことを確認する．残っていたら NEW を実行する．

```
LIST
10 REM MAENO
20 REM PROGRAM
Ok
NEW
Ok
LIST
Ok
```

28．トラブルシューティング

3．プログラム実行時のデバッグ

RUN してエラーが出たら，まず巻末資料のエラーメッセージ一覧を見て，その意味を確認し，原因を考えてからプログラムを修正してください．
それでもわからないときは，次のような点に注意してみてください．

Syntax error

● 2つの行がつながっていないか

まず LIST を実行し，カーソルを一番上の行の行番号の上に移動し，一回ずつ RETURN キーを押してみてください．このとき，カーソルは次の行番号の上に移動します．もし，ある行の行番号をとばしてしまうようなら，その行は前の行に含まれていることになります．

```
10 PRINT "A"
20 PRINT "B"
30 PRINT "C"
```

カーソルが10行にあるとき RETURN キーを押したのにカーソルは30行にいってしまう．

```
LIST 10
10 PRINT "A"
20 PRINT "B"
Ok
```

LIST 10 としてみると……
20行らしきものがくっついている．

```
10 PRINT "A" □
```

カーソルを10行の終りに移動して CTRL + E を入力するとこのゴミは消える．このあと，続けて RETURN キーを押す．

```
20 PRINT "B"
```

20行をもう1度打ち直して RETURN ．

これでOK．

● 見逃しやすい書き違い．

"I" と "1"，"O" と "0"，"；" と "："，カッコの対応 "(" と ")"，"，" と "．"，パラメータの数など．COST，TANK など予約語を含む変数名は使えない．

予約語 P15

28. トラブルシューティング

● エラー発生行には Syntax error がない場合

```
10 READ N
20 PRINT N
30 DATA C
```

Syntax error in 30 となるが10行を 10 READ N$ とするべき場合もある.

```
10 DEFFNK(L, N)=2/(L, N)
20 B=FNK(2, 4)
30 PRINT B
```

Syntax error in 20 となるが，実際には10行が間違っており，右辺を計算式にしなければなりません.

Illegal function call

パラメータ

● エラーがでた行に原因があるとは限らない．その行にコマンドや関数のパラメータとして使っている変数があったら，値を PRINT してみる．予想外の値になっていたらなぜそうなったかをつきとめる．

```
10 FOR I=0 TO 255
20 C=!+1 MOD 16
30 COLOR C
40 NEXT
RUN
Illegal function call in 30
Ok
?C
 16
Ok
LIST 20
20 C= (I+1) MOD 16 (修正する)
Ok
RUN
Ok
```

● RENUM で行の順を変えようとしたり，SCREEN 1 で PSET を使うなどは許されない．BASIC 説明書で調べ，関数の使い方は実験しておく．

28．トラブルシューティング

許されるパラメータの範囲

0 ～255

ERROR, LOCATE, ON GOTO や ON GOSUB の飛び先を決める式の値, OUT, POKE や VPOKE のデータ, CHR$, EOF, INP, INPUT$, INSTR, LEFT$, MID$, RIGHT$, SPACE$, STRING$

TAB

その他

BASE 0～19, CLEAR 0～, ～&HF380, COLOR 0～15

KEY 1～10, MAXFILES 0～15, PAD 0～7, PDL 1～12

PLAY 0～3, SPRITE$ 0～255, SCREEN 0～3, SOUND 0～13, 0～255

STICK 0～2, STRIG 0～4, VDP 0～8, VPEEK 0～16383

VPOKE 0～16383, WIDTH 1～32 (SCREEN1) または1～40 (SCREEN0)

配列の添字や SQR の引数は正または 0，LOG の引数は正でなければならない．

添字

4. エラーメッセージのでないバグの発見

同じバグでもエラーメッセージはでずに実行結果だけがおかしいものもあります．

- 無限ループなどプログラムの流れがおかしい．

無限ループ

CTRL + STOP キーを押してみるといつも同じようなところを実行し，先へ進んでいない．TRON コマンドを実行してみると実行している行が予想外である．

```
10 FOR I=65 TO 255
20 PRINT CHR$(I)
30 GOSUB 100
40 NEXT
50 END
100 FOR I=0 TO 100
110 NEXT
120 RETURN
```

28. トラブルシューティング

FOR〜NEXT で使う変数の重複，不用意な変更で予想外の結果になることがよくあるので気をつける．

ON〜GOTO (GOSUB) 文でその変数が 0，または飛び先の数以上のときは，次のステートメントに制御が移る．

```
10 INPUT A
20 ON A GOTO 30, 40, 50
30 PRINT 1 : END
40 PRINT 2 : END
50 PRINT 3
60 END
```

Aが 0 や 4 以上のとき30行が実行されてしまう．

ヌル
ストリング

INSTR を使うときは ″″（ヌルストリング）に注意．

```
10 ON INSTR ("ABC" , INKEY$) GOTO 30, 40, 50
20 GOTO 10
30 PRINT "A" : END
40 PRINT "B" : END
50 PRINT "C"
60 END
```

これは，RUN 直後 INKEY$の値として″″が返され，INSTR(″ABC″, ″″）の値は 1 となりすぐに30行が実行されてしまう．10〜20行は

```
10 A$=INKEY$: IF A$= "" THEN 10
20 ON INSTR ("ABC" , A$) GOTO 30, 40, 50: GOTO 10
```

のように工夫する．

- 数値がおかしい

演算の
優先順位

変数名
P 31

演算の優先順位の誤解，変数名の重複などが原因．

```
10 PRICE=10
20 PROGRAM=20
30 PRINT PRICE * PROGRAM
RUN
  400
Ok
```

28．トラブルシューティング

PRICE と PROGRAM は同じ変数名（頭 2 文字で区別される）になってしまう．

```
10 DEFINT A-Z
20 A=SIN (0.5)
30 PRINT A
RUN
  0
Ok
```

SIN の答は－1～＋1．整数型の変数では，これらはすべて 0 になってしまうので実数型を使う．

```
?10-4 MOD 3
  9
Ok
? (10－4) MOD 3
  0
Ok
?10／3 ＊ 3
  9.9999999999999
Ok
```

なども意識しておくこと．

● 画面表示

それぞれの SCREEN で何文字，何ドット表示できるか確認すること．テキスト画面では，最下行に何かを PRINT すると全体がスクロールしてしまう．セミコロンを使えばこれを防げるが，最下行の右端に PRINT するときはスクロールは避けられない．

スクロール

数値や文字列を並べて表示するときは，PRINT USING 文を使う．〝；″，〝，″，TAB，SPC，SPACE$，STR$，なども使いこなせるよう，よく実験しておくこと．
HEX$，OCT$，BIN$などのケタをそろえるには，RIGHT$を使う．

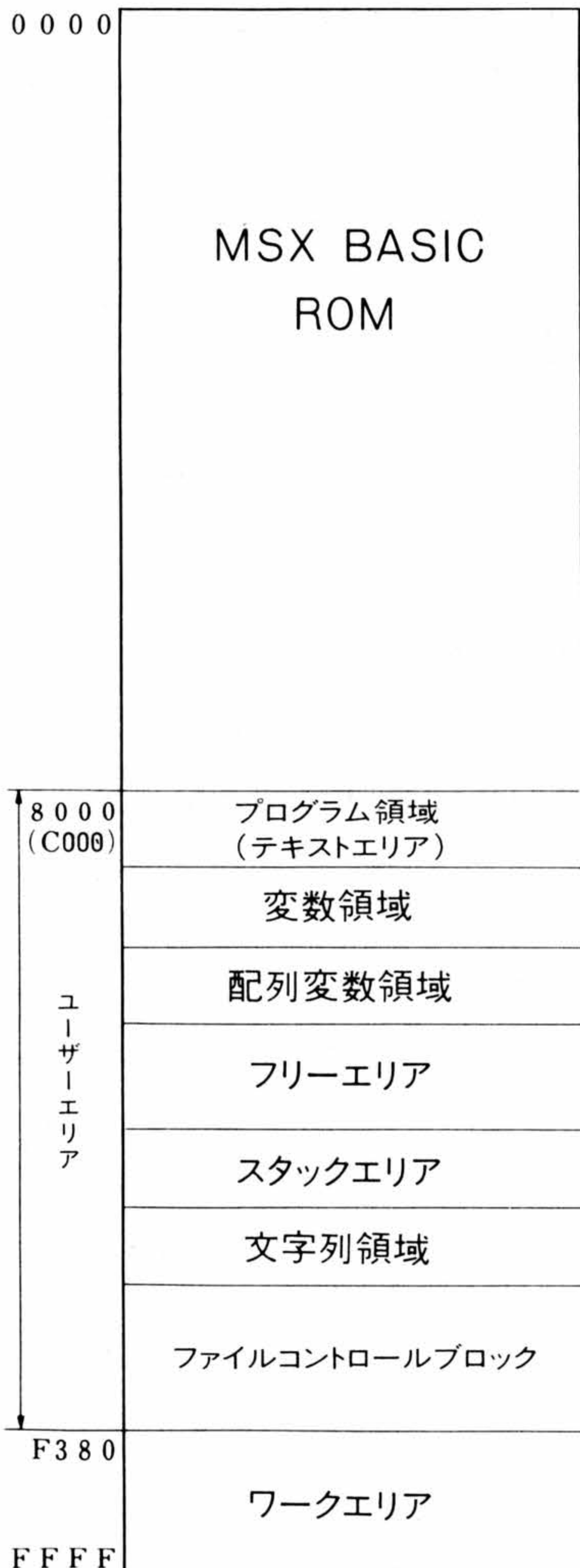
```
10 FOR I=1 TO 8
20 R=RND (1) ＊ 255
30 PRINT RIGHT$ (〝00000000″+BIN$ (R) ,8)
40 NEXT
```

3 資料編

1. 主要LSIとメモリ

- **CPU** Z-80A相当品
  クロック 3.579545MHz（テレビ色副搬送波周波数）
  M1サイクルで1WAIT入れる
- **CRTC** TMS-9918A 相当品
- **PSG** AY-3-8910 相当品
- **PPI** I-8255 相当品
- **ROM** MSX BASIC インタプリタ 32KB
- **RAM** 16KB標準

- 本体内の四つの基本スロットで，ROM・RAM合わせて256KBまで収容可能です．また，拡張ユニット一つごとに四スロット・256KB収容できます．
- BASIC ROMは0番地から7FFF番地に，またRAMはFFFF番地より若い番地に向かって実装します．詳細はメモリマップを参照してください．

2. メモリマップ

ユーザーエリアは16KRAM実装の場合$(C000)_{16}$〜$(F37F)_{16}$, また, 32KRAM実装の場合$(8000)_{16}$〜$(F37F)_{16}$です.

3. I/Oマップ

① I/Oアドレス割り当て

アドレス	割り当て
00	
80	＊RS-232C
90	＊プリンタ
98	VDP
A0	PSG
A8	PPI
B0	
C0	
D0	＊FDC
D8	＊漢字ROM
E0	
FF	

IO ADR	RW	内　容	備　考
&H98	W	V-RAMへのデータライト	9918A
	R	V-RAMからのデータリード	相当品
&H99	W	コマンド，アドレスセット	
	R	ステータスリード	
&HA0	W	アドレスラッチ	AY-3-8910
&HA1	W	データライト	相当品
&HA2	R	データリード	
&HA8	W	ポートAデータライト	8255A
	R	ポートAデータリード	相当品
&HA9	W	ポートBデータライト	
	R	ポートBデータリード	
&HAA	W	ポートCデータライト	
	R	ポートCデータリード	
&HAB	W	モードセット	
&H90	W	ストローブ出力（b0）	ラッチ出力
	R	ステータス入力（b1）	BUSYで'1'
&H91	W	プリントデータ	ラッチ出力

I/Oアドレス80番地からFF番地はシステム用として上記のように定めてあります。空欄はシステムリザーブです。

＊印の入出力番地はオプション機器のために用意されていることを意味します。

② PPIビット割り当て

ポート	ビット	I／O	信号名	内　容
A	0 1	出力	CS0L CS0H	０００００−３ＦＦＦ番地の スロット指定番号
	2 3		CS1L CS1H	４０００−７ＦＦＦ番地の スロット指定信号
	4 5		CS2L CS2H	８０００−ＢＦＦＦ番地の スロット指定信号
	6 7		CS3L CS3H	Ｃ０００−ＦＦＦＦ番地の スロット指定信号
B	0 | 7	入力		キーボードリターン信号
C	0 1 2 3	出力	KB0 KB1 KB2 KB3	キーボードスキャン信号
	4		CASON	カセットコントロール(L-ON)
	5		CASW	カセット書き込み信号
	6		CAPS	CAPSランプ信号(Lで点燈)
	7		SOUND	ソフトによるサウンド出力

③PSGビット割り当て

ポート	ビット	I／O	接続コネクターピン番号		ジョイスティック使用時の信号
A	0	↑	J3—1ピン	*1	FWD1
			J4—1ピン	*2	FWD2
	1		J3—2ピン	*1	BACK1
			J4—2ピン	*2	BACK2
	2		J3—3ピン	*1	LEFT1
			J4—3ピン	*2	LEFT2
	3		J3—4ピン	*1	RIGHT1
		入力	J4—4ピン	*2	RIGHT2
	4		J3—6ピン	*1	TRGA1
			J4—6ピン	*2	TRGA2
	5		J3—7ピン	*1	TRGB1
			J4—7ピン	*2	TRGB2
	6		キー配列指定入力	*4	"H"／"L"レベル
	7	↓	CSAR(カセットテープのリード)		
B	0	↑	J3—6ピン	*3	"H"レベル
	1		J3—7ピン	*3	
	2		J4—6ピン	*3	
	3		J4—7ピン	*3	
	4	出力	J3—8ピン		
	5		J4—8ピン		
	6		ポートA入力セレクト		
	7	↓	KLAMP(カナランプ信号"L"で点燈)		

*1 ポートBのビット6が"L"レベル時有効。ジョイスティック1用

*2 ポートBのビット6が"H"レベル時有効．ジョイスティック2用

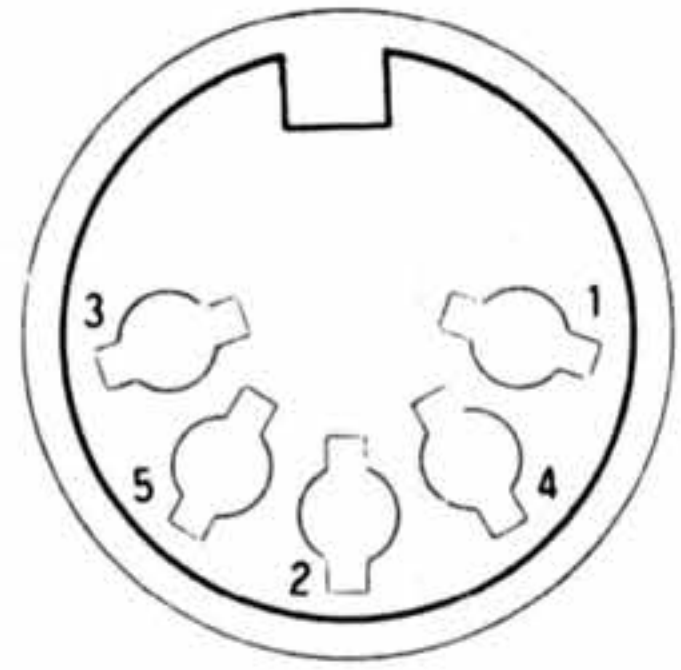
*3 出力ポートとして使用しない時は"H"レベルにしてください．

*4 JIS配列——"H"レベル，あいうえお配列——"L"レベル

4. コネクター一覧

端子名	仕様／規格
ビデオ出力 複合映像信号	DIN　5ピンコネクタ RCA　2ピンコネクタ
カセット	DIN　8ピンコネクタDIN-45326
ジョイスティック端子	AMP　9ピンコネクタ
カートリッジバス	2.54ピッチ,50ピンコネクタ
オーディオ出力	RCA 2ピンコネクタ

① DIN 5 ピンコネクタ信号線表

接続端子	信号名
1	＋5 V
2	GND
3	AUDIO
4	MONITOR VIDEO
5	RF VIDEO

② カセットインターフェイス

○入　　　　力：カセット側のイヤホン端子に接続

○出　　　　力：カセット側のマイクロホン端子に接続

○同　期　方　式：調歩同期(非同期)方式

○転　送　レ　ー　ト：1200baud(1200Hz-1波"0",2400Hz-2波"1")

デフォルト

2400baud(2400Hz-1波"0",4800Hz-2波"1")

ソフトウェア切り替え

(2400baudは専用データレコーダにおいてのみ使用可)

○変　調　方　式：FSK方式

○リモート機能：有り

○コ　ネ　ク　タ：DIN 8ピン

○信 号 線 表

端子番号	信号名
1	GND
2	GND
3	GND
4	CMTOUT
5	CMTIN
6	REM＋
7	REM－
8	GND

ピンコネクション

③ ジョイスティック端子

○使用ＩＣ：AY-3-8910相当品

○入 出 力：入力4bit，出力1bit，双方向2bit（1ポートにつき）

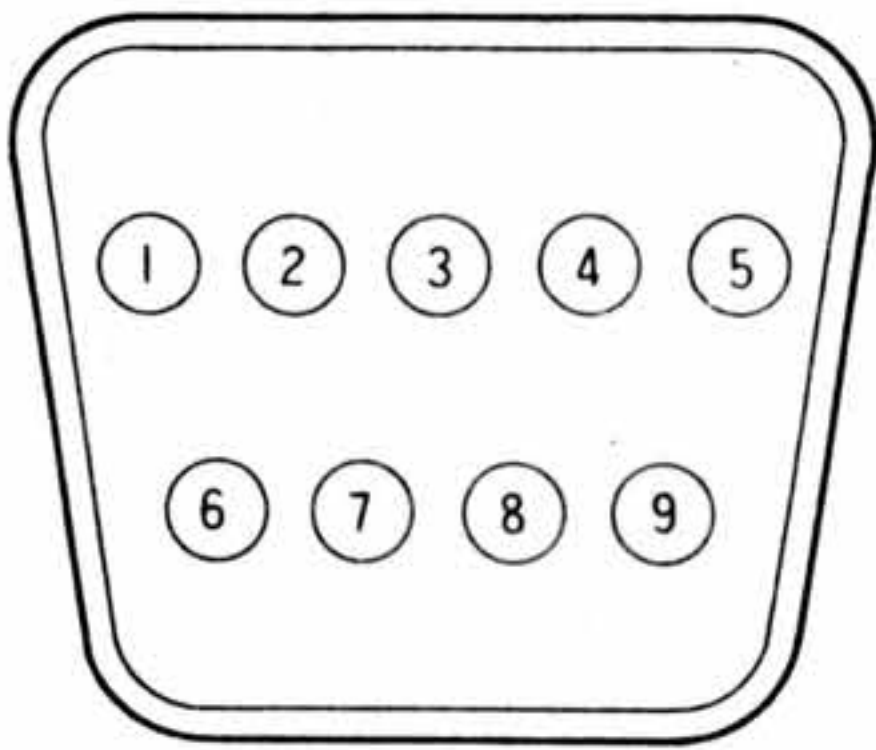
○ロジック：正論理

○レ ベ ル：TTL

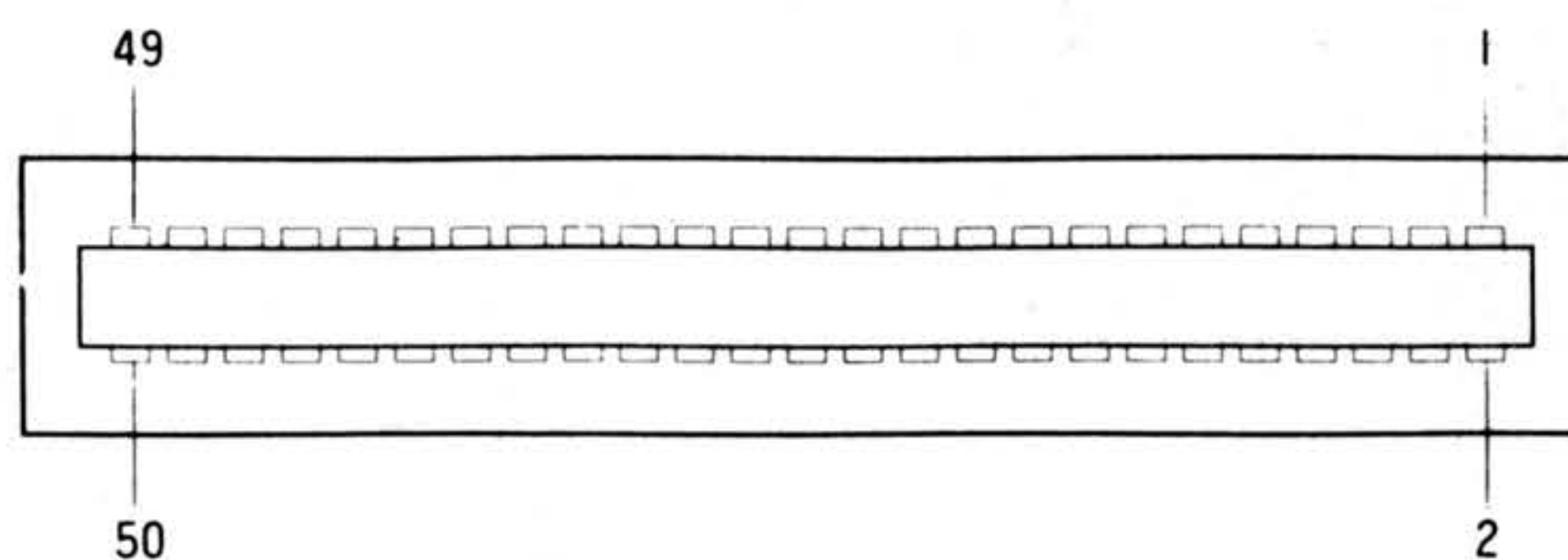
○コネクタ：AMP 9ピン

○信号線表

端子番号	信号名
1	FWD
2	BACK
3	LEFT
4	RIGHT
5	＋5V
6	TRG 1
7	TRG 2
8	出力
9	GND

ピンコネクション

④ カートリッジバス　信号線表

ピンNo.	名　称	I/O *	ピンNo	名　称	I/O *
1	$\overline{\mathrm{CS1}}$	O	2	$\overline{\mathrm{CS2}}$	O
3	$\overline{\mathrm{CS12}}$	O	4	$\overline{\mathrm{SLTSL}}$	O
5	予備　注(1)	—	6	$\overline{\mathrm{RFSH}}$	O
7	$\overline{\mathrm{WAIT}}$　注(2)	I	8	$\overline{\mathrm{INT}}$　注(2)	I
9	$\overline{\mathrm{M1}}$	O	10	$\overline{\mathrm{BUSDIR}}$	I
11	$\overline{\mathrm{IORQ}}$	O	12	$\overline{\mathrm{MERQ}}$	O
13	$\overline{\mathrm{WR}}$	O	14	$\overline{\mathrm{RD}}$	O
15	$\overline{\mathrm{RESET}}$	O	16	予備・注(1)	—
17	A 9	O	18	A15	O
19	A11	O	20	A10	O
21	A 7	O	22	A 6	O
23	A12	O	24	A 8	O
25	A14	O	26	A13	O
27	A 1	O	28	A 0	O
29	A 3	O	30	A 2	O
31	A 5	O	32	A 4	O
33	D 1	I/O	34	D 0	I/O
35	D 3	I/O	36	D 2	I/O
37	D 5	I/O	38	D 4	I/O
39	D 7	I/O	40	D 6	I/O
41	GND	—	42	CLOCK	O
43	GND	—	44	SW1	—
45	＋5 V	—	46	SW2	—
47	＋5 V	—	48	＋12V	—
49	SUNDIN	I	50	−12V	—

* 本体を基準にした入出力の区別

注(1)　予備は使用禁止端子

注(2)　OPEN COLLECTER 出力

カートリッジバス　信号線説明

ピンNo.	名　称	内　　　容
1	$\overline{CS1}$	ROM ４０００～７ＦＦＦ番地セレクト信号
2	$\overline{CS2}$	ROM ８０００～ＢＦＦＦ番地セレクト信号
3	$\overline{CS12}$	ROM ４０００～ＢＦＦＦ番地セレクト信号(256KROM用)
4	$\overline{SLTSL}$	スロットセレクト信号．各スロット毎にそのスロット固有のセレクト信号を印加する．
5	予備	将来のための予備信号線 — 使用禁止
6	$\overline{RFSH}$	リフレッシュサイクル信号
7	$\overline{WAIT}$	CPUのWAIT要求信号
8	$\overline{INT}$	CPUへの割込み要求信号
9	$\overline{M1}$	CPUのフェッチサイクルを表わす記号
10	$\overline{BUSDIR}$	外部データバスバッファの方向を制御する信号 カートリッジがセレクトされ，データを送出するタイミングでメモリを除く各カードリッジよりＬレベルを出力する．
11	$\overline{IORQ}$	I／Oリクエスト信号
12	$\overline{MERQ}$	メモリリクエスト信号
13	$\overline{WR}$	ライトタイミング信号
14	$\overline{RD}$	リードタイミング信号
15	$\overline{RESET}$	システムリセット信号
16	予備	将来のための予備信号線 — 使用禁止
17～32	A0～A15	アドレスバス信号
33～40	D0～D7	データバス信号
41	GND	信号グランド
42	CLOCK	CPUクロック 3.579545MHz
43	GND	信号グランド
44, 46	SW1, SW2	抜差プロテクト用
45, 47	＋５Ｖ	＋５Ｖ電源
48	＋12V	＋12V電源
49	SUNDIN	サウンド入力信号(—5bdm)
50	－12V	—12V電源

5.　オプション一覧

ジョイスティック
タブレット
カセットテープレコーダ
コンパクトフロッピーディスク
プロッタプリンタ
プリンタ
RS-232Cアダプタ
RFコンバータ
16KB RAMカートリッジ
拡張ユニット

など周辺機器を順次準備しております。

6. エラーメッセージ一覧

エラーメッセージ	エラーコード
Bad file name	56

ファイルの構造がおかしい。

- 適当でないファイル名でOPENしようとした。
- 出力専用のファイルから入力しようとした。または，その逆。

Bad file number 52

ファイル番号がおかしい。

- OPENしていないファイル番号でPRINT #などを実行した。
- MAXFILES文で定義せずに，2～15のファイル番号を使った。

Can't CONTINUE 17

プログラムの実行が再開できない。

- エラーの発生による実行の中断の場合。
- 中断させた後にプログラムを変更した場合。

Device I/O error 19

周辺装置との入出力で，エラーが発生した。

- カセットテープにキズがある。
- カセットレコーダのレベル調整が合っていない。
- プリンタ，カセットレコーダなどの電源が切れ，コマンドが中断されたとき。

Direct statement in file 57

アスキー型式のプログラムファイルが正しくない。

- プログラムファイル中に行番号のないステートメントがあった。
- ロードしようとしたファイルがBASICプログラム以外のテキストである。

Division by zero 11

0で除算を行った。

- 除数が0になっていた。
- 定義していない変数で除算を行った。

エラーメッセージ	エラーコード

FIELD overflow　50

将来の拡張のために用意されたエラー.

- FIELD 文で256バイト以上の大きさの領域を指定した.

File already open　54

すでにファイルがOPEN されている.

- CLOSE を実行せずに同じファイル番号を使って OPEN した.

File not OPEN　59

ファイルがOPEN されていない.

- PRINT #, INPUT #などで使うファイル番号が, まだ OPEN 文によって定義されていない.

File not found　53

指定した名前のファイルが存在しない. 将来の拡張(ディスク)のためのエラー.

- ファイル名の書き間違い.
- 拡張子が違っている.

Illegal direct　12

ダイレクト・モードで実行できないステートメントを実行しようとした.

- DEFFN 文はダイレクト・モードでは使えない.

Illegal function call　5

ステートメントや関数の呼び方が間違っている.

- ステートメントや関数のパラメータが規定の範囲を超えている.
- 配列の添字の値が負になっている.

Input past end　55

ファイルのデータを読み尽した.

- INPUT #, LINEINPUT #を実行する回数がファイル中のデータ数より多い.

エラーメッセージ	エラーコード

Internal error　51

インタープリタ内に異常が発生した．

- POKE 文，USR 関数を誤って使うと発生することがある．

Line buffer overflow　25

入力ラインバッファが一杯になった．

- 1行で入力できるのは255文字以内．

Missing operand　24

必要なパラメータが無い．

- パラメータの数を間違えた．
- 数値間のカンマがピリオドになっている．

NEXT without FOR　1

FOR 文が足りない．

- 入れ子構造が正しくない．
- FOR～NEXT 文のループの中に他から飛び込んでしまった．

No RESUME　21

エラー処理ルーチンに RESUME 文がない．

- エラー処理ルーチンの終わりは，END 文，RESUME 文，ON ERROR GOTO 0 文のいずれかの文でなければならない．
- エラー処理ルーチンから戻る場合に GOTO 文を使った．

Out of DATA　4

READ 文で読むべきデータがない．

- データの数が不足．
- RESTORE 文が間違っている．
- DATA 文の区切り記号の使い方が間違っている．

エラーメッセージ	エラーコード
Out of memory	**7**

メモリ容量が足りない.

- プログラムが長すぎる.
- 変数を多く使いすぎる.
- 配列が大きすぎる.
- FOR～NEXT 文，GOSUB～RETURN 文のネスティングが深すぎる.
- CLEAR 文で領域の設定が大きすぎる.

Out of string space	**14**

文字領域が足りない.

- CLEAR 文で設定する文字領域が小さい.

Overflow	**6**

数値が許される範囲を超えている.

- 整数演算の結果が−32768～32767の範囲を超えた.
- 実数演算の結果が$-9.\underbrace{99\cdots99}_{\text{14個}}$ E62～$9.\underbrace{99\cdots99}_{\text{14個}}$ E62の範囲を超えた.
- アドレスをパラメータとする命令で，指定された値が範囲を超えた.

RESUME without error	**22**

エラーが発生していないのに RESUME 文を実行した.

- エラー処理ルーチンへ GOTO 文，GOSUB 文で分岐している.
- メインルーチンの最後の行の END 文がなく，すぐ後のエラー処理ルーチンを実行してしまった.

RETURN without GOSUB	**3**

GOSUB 文の実行以前に RETURN 文に出合った.

- サブルーチンへ GOTO 文で分岐している.
- メインルーチンの最後の行に END 文がなく，すぐ後のサブルーチンを実行してしまった.

エラーメッセージ	エラーコード

Redimensioned array　10

配列を2重に定義した.

- ERASE文を忘れ，同じ名前の配列を宣言した.
- 宣言をせずに配列変数を使い，その後DIM文で宣言した.
- ループの中にDIM文がある.

Sequential I/O only　58

将来の拡張のために用意されたエラー.

- シーケンシャル以外のファイル入出力はできない.

String formula too complex　16

文字式が複雑すぎる.

- 1行に書かれた文字列の演算が複雑すぎる. カッコの重なり(ネスティング)が深すぎる.

String too long　15

文字列が長すぎる.

- 文字変数に256文字以上代入しようとした.

Subscript out of range　9

配列の添字が許される範囲を超えている.

- 添字値が大きすぎる.
- 未定義の配列で添字が11以上である.

Syntax error　2

MSX BASICの文法にそっていない.

- 規定のキーワード（予約語）以外のものがある.
- カッコが正しく対応していない.
- 区切りが間違っている（カンマ，ピリオド，コロン，セミコロン等).
- 変数名が英字で始まっていない.

- 変数名がキーワード（予約語）を含んでいる．
- 関数，ステートメント等のパラメータの数がおかしい．
- プログラムの入力ミス．

Type mismatch　13

変数の型があわない．

- 数値変数に文字を代入しようとした．
- 文字変数に数値を代入しようとした．
- 関数の引数の型があっていない．

Undefined line number　8

行番号の指定がおかしい．

- 行番号の指定が行われていない．
- GOTO 文，GOSUB 文の分岐先，RESTORE 文，RUN で指定した行番号が存在しない．

Undefined user function　18

ユーザー関数が定義されていない．

- 変数の名前に FN で始まるものが使われている．
- DEFFN 文の関数名を間違えた．
- DEFFN 文を実行していない（GOTO 文などでプログラムを途中から実行した）．

Unprintable error　23，26～49，60～255

コード表以外のエラー・コードを持つエラーが発生した．

- ERROR 文を実行した．

Verify error　20

カセットテープ上のプログラムファイルのベリファイに失敗した．

- CLOAD? コマンドでベリファイしたが，テープ上のプログラムとテキストエリアのプログラムが異なっている．

7. コントロールコード表

コード(10進)	コード(16進)	機　　能	対応キー
0	00		
1	01	グラフィックキャラクタの入出力時のヘッダ	CTRL + A
2	02	カーソルを直前の語の先頭へ移動	CTRL + B
3	03	入力待ち状態を終了する	CTRL + C
4	04		CTRL + D
5	05	カーソル以下を削除	CTRL + E
6	06	カーソルを次の語の先頭へ移動	CTRL + F
7	07	スピーカを鳴らす(BEEP文と同じ)	CTRL + G
8	08	カーソルの1つ前の文字を削除する	CTRL + H , BS
9	09	次の水平タブ位置へ移動	CTRL + I , TAB
10	0A	行送り(ラインフィード)	CTRL + J
11	0B	カーソルをホームポジション(左上)に戻す	CTRL + K
12	0C	画面をクリアし、カーソルをホームポジションに戻す	CTRL + L
13	0D	カーソルを左端に戻す(キャリッジリターン)	CTRL + M , RETURN
14	0E	カーソルを行末へ移動	CTRL + N
15	0F		CTRL + O
16	10		CTRL + P
17	11		CTRL + Q
18	12	挿入モードのON/OFFスイッチ	CTRL + R , INS
19	13		CTRL + S
20	14		CTRL + T
21	15	一行を画面から削除	CTRL + U
22	16		CTRL + V
23	17		CTRL + W
24	18		CTRL + X , SELECT
25	19		CTRL + Y
26	1A		CTRL + Z
27	1B		CTRL + [, ESC
28	1C	カーソルを右へ移動	CTRL + ¥ , →
29	1D	カーソルを左へ移動	CTRL +] , ←
30	1E	カーソルを上へ移動	CTRL + ^ , ↑
31	1F	カーソルを下へ移動	CTRL + _ , ↓
127	7F	カーソルの指す文字を削除	DEL

8. キャラクタコード表

コード（10進）	コード（16進）	キャラクタ	コード（10進）	コード（16進）	キャラクタ	コード（10進）	コード（16進）	キャラクタ	コード（10進）	コード（16進）	キャラクタ
0	00	↑	32	20		64	40	@	96	60	
1	01		33	21	!	65	41	A	97	61	a
2	02		34	22	"	66	42	B	98	62	b
3	03		35	23	#	67	43	C	99	63	c
4	04		36	24	$	68	44	D	100	64	d
5	05		37	25	%	69	45	E	101	65	e
6	06		38	26	&	70	46	F	102	66	f
7	07		39	27	'	71	47	G	103	67	g
8	08	コ	40	28	(	72	48	H	104	68	h
9	09		41	29	)	73	49	I	105	69	i
10	0A	ン	42	2A	*	74	4A	J	106	6A	j
11	0B	ト	43	2B	+	75	4B	K	107	6B	k
12	0C		44	2C	,	76	4C	L	108	6C	l
13	0D	ロ	45	2D	-	77	4D	M	109	6D	m
14	0E	ー	46	2E	.	78	4E	N	110	6E	n
15	0F		47	2F	/	79	4F	O	111	6F	o
16	10	ル	48	30	0	80	50	P	112	70	p
17	11	キ	49	31	1	81	51	Q	113	71	q
18	12	ャ	50	32	2	82	52	R	114	72	r
19	13		51	33	3	83	53	S	115	73	s
20	14	ラ	52	34	4	84	54	T	116	74	t
21	15	ク	53	35	5	85	55	U	117	75	u
22	16	タ	54	36	6	86	56	V	118	76	v
23	17		55	37	7	87	57	W	119	77	w
24	18		56	38	8	88	58	X	120	78	x
25	19		57	39	9	89	59	Y	121	79	y
26	1A		58	3A	:	90	5A	Z	122	7A	z
27	1B		59	3B	;	91	5B	[	123	7B	{
28	1C		60	3C	<	92	5C	¥	124	7C	|
29	1D		61	3D	=	93	5D	]	125	7D	}
30	1E		62	3E	>	94	5E	^	126	7E	~
31	1F	↓	63	3F	?	95	5F	—	127	7F	

コード（10進）	コード（16進）	キャラクタ	コード（10進）	コード（16進）	キャラクタ	コード（10進）	コード（16進）	キャラクタ	コード（10進）	コード（16進）	キャラクタ
128	80	♠	160	A0		192	C0	タ	224	E0	た
129	81	♥	161	A1	。	193	C1	チ	225	E1	ち
130	82	♣	162	A2	「	194	C2	ツ	226	E2	つ
131	83	◆	163	A3	」	195	C3	テ	227	E3	て
132	84	○	164	A4	、	196	C4	ト	228	E4	と
133	85	●	165	A5	・	197	C5	ナ	229	E5	な
134	86	を	166	A6	ヲ	198	C6	ニ	230	E6	に
135	87	ぁ	167	A7	ァ	199	C7	ヌ	231	E7	ぬ
136	88	ぃ	168	A8	ィ	200	C8	ネ	232	E8	ね
137	89	ぅ	169	A9	ゥ	201	C9	ノ	233	E9	の
138	8A	ぇ	170	AA	ェ	202	CA	ハ	234	EA	は
139	8B	ぉ	171	AB	ォ	203	CB	ヒ	235	EB	ひ
140	8C	ゃ	172	AC	ャ	204	CC	フ	236	EC	ふ
141	8D	ゅ	173	AD	ュ	205	CD	ヘ	237	ED	へ
142	8E	ょ	174	AE	ョ	206	CE	ホ	238	EE	ほ
143	8F	っ	175	AF	ッ	207	CF	マ	239	EF	ま
144	90		176	B0	ー	208	D0	ミ	240	F0	み
145	91	あ	177	B1	ア	209	D1	ム	241	F1	む
146	92	い	178	B2	イ	210	D2	メ	242	F2	め
147	93	う	179	B3	ウ	211	D3	モ	243	F3	も
148	94	え	180	B4	エ	212	D4	ヤ	244	F4	や
149	95	お	181	B5	オ	213	D5	ユ	245	F5	ゆ
150	96	か	182	B6	カ	214	D6	ヨ	246	F6	よ
151	97	き	183	B7	キ	215	D7	ラ	247	F7	ら
152	98	く	184	B8	ク	216	D8	リ	248	F8	り
153	99	け	185	B9	ケ	217	D9	ル	249	F9	る
154	9A	こ	186	BA	コ	218	DA	レ	250	FA	れ
155	9B	さ	187	BB	サ	219	DB	ロ	251	FB	ろ
156	9C	し	188	BC	シ	220	DC	ワ	252	FC	わ
157	9D	す	189	BD	ス	221	DD	ン	253	FD	ん
158	9E	せ	190	BE	セ	222	DE	゛	254	FE	
159	9F	そ	191	BF	ソ	223	DF	゜	255	FF	

グラフィックキャラクタコード表

コード(10進)	コード(16進)	キャラクタ	コード(10進)	コード(16進)	キャラクタ
0	00		16	10	π
1	01	月	17	11	┴
2	02	火	18	12	┬
3	03	水	19	13	┤
4	04	木	20	14	├
5	05	金	21	15	┼
6	06	土	22	16	│
7	07	日	23	17	─
8	08	年	24	18	┌
9	09	円	25	19	┐
10	0A	時	26	1A	└
11	0B	分	27	1B	┘
12	0C	秒	28	1C	╳
13	0D	百	29	1D	大
14	0E	千	30	1E	中
15	0F	万	31	1F	小

ここに掲げるグラフィックキャラクタをキーボードから入力するとCHR$(1)+CHR$(グラフィックキャラクタコード+64)の2バイトが入力されます.また,PRINT文やLPRINT文でグラフィックキャラクタを出力すると,(例:PRINT"月")同じように,CHR$(1)+CHR$(グラフィックキャラクタコード+64)の2バイトが自動的に出力されます.ここで1文字目のCHR$(1)は,後の文字がグラフィックキャラクタであることを表すコントロールコードです.CHR$関数を使ってグラフィックキャラクタを出力する場合はCHR$(1)+CHR$(グラフィックキャラクタコード+64)で出力してください.

9. 用語集

● BASIC

コンピュータのプログラムを書くための言語の一つ.

● BCD

Binary Coded Decimal. 2 進化10進数. 4 bit で10進数 1 ケタを表す.

● CPU

Central Processing Unit.中央処理装置.

● CRT

Cathode Ray Tube. テレビ. ブラウン管のこと.

● FAC

Floating Accumulator Counter.メモリ上に想定されたレジスタ群. 浮動小数アキュムレータ.

● I/O ポート

CPU が周辺機器と信号を入出力するところ.

● LSB

Least Significant Bit.最下位（第 0 ）ビット.

● LSI

Large Scale Integration. 大規模集積回路.

● MSB

Most Significant Bit.最上位ビット. 1 バイトなら第 7 , 2 バイト 1 組なら第15ビットがこれにあたる.

● PSG

Programmable Sound Generator.音声を作るための専用 CPU. MSX ではサブ CPU として使われる.

● RAM

Random Access Memory.内容を随時書きかえられるメモリ.

● **RFコンバータ**

Radio Frequency Converter. 家庭用テレビ受像機にコンピュータの信号を映像として出すための装置.

● **ROM**

Read Only Memory.読み出し専用メモリ. 一度内容を書き込むと二度と書きかえることができない.

● **RS232C回線**

データを伝送するケーブル. タイミング, データ形式, 伝送速度などの規格がある.

● **VDP**

Video Display Processor.映像（ビデオ信号として）操作専用のCPU. MSXではサブCPUとして使われる.

● **アキュムレータ**

加算, 減算などの演算を行うレジスタ. Z-80では通常Aレジスタのこと. これと同様の働きをするものを一般にアキュムレータと呼ぶこともある.

● **アスキー形式**

ASCII（American Standard Code for Information Interchange）で定めたコードに従ってデータを伝送, 蓄積する形式.

● **アセンブラ**

Assembler.ニーモニックで書かれたプログラムを機械語に翻訳するプログラム.

● **アドレス**

Address →番地.

● **アナログ**

Analog.連続量. 例えば0と1の間には無限の小数があり, 0から1まではつながっているという考え方.

● **インターフェイス**

Interface.異なった物の間でのデータをやりとりするための仲介役となる装置,回路など.

● **インタープリタ**

Interpreter. BASICインタープリタはBASICで書かれたプログラムを機械語に翻訳しながら実行するプログラムで，それ自身は機械語で書かれている．

● **インターラプト**

Interrupt.→割込み

● **エディタ**

Editor.インタープリタプログラムの中で，テキストを作成するための部分．

● **エラー**

Error.プログラムやコマンドを実行中に，実行不可能な命令があったり，外部的にそのような状態が引き起こされた場合に発生するもの．

● **エラートラップ**

Error Trap.エラーを検出すること，またはその機能．BASICにはあるが機械語にはない．

● **演算子**

BASICでは，(,)，^，−(負号)，*，／，¥，MOD，+，−，<，>，=，NOT，AND，OR，XOR，IMP，EQVを使う．基本的な計算に使う特殊記号のこと．

● **演算の優先順位**

カッコ内のもの，関数，指数，負号，乗除算，加減算，関係演算，論理演算の順．同位の場合は左側が優先．

● **エンベロープ**

Envelope.包絡線．SOUND文でいうエンベロープは，実際の音波をPSGで扱うために加工した波形のこと．

● **オートリピート**

Auto Repeat.一つのキーを押し続けると，連続的にそのキーが入力できる機能．

● **オーバーフロー**

Overflow.数値がBASICで扱える数値の範囲を超えてしまうこと．桁あふれ．

● **オプション**

Option.標準システムに追加できる周辺装置．コマンドに追加できるパラメータを指すこともある．

● **カーソル**

Cursor.現在キー入力したものが，どこに表示されるかを示す画面上の■や_のこと．

● **外部記憶装置（媒体）**

コンピュータ内部のデータを外部に保存しておくための媒体．通常カセットテープやディスクのこと．

● **拡張子**

ファイル名などに追加できる文字列など．標準 MSX BASIC では扱わない．

● **格納**

メモリ内に記憶すること．

● **画素**

画面を扱う上での最小単位．通常 1 ドットのこと．

● **仮想画面**

実際にはテレビモニタには 1 つの画面しかないが，そこに何枚もの画面が重なっているものとして扱う場合の画面．

● **型宣言**

変数の型が何型かをプログラムの始めの方で定義してしまうこと．

● **型変換**

型の違う定数や変数の間で演算や代行を行うとき，片方の型の値を四捨五入，切り捨てなどの操作で合う型に換えてしまうこと．

● **ガベージコレクション**

Garbage Collection.文字列領域が足りなくなったときなどに，不要になった文字列を整理して領域を有効に使おうとする BASIC インタープリンタの動作．この動作中は STOP キーも無効となる．

● **画面構成**

MSX BASICは4つの画面モードをもち，(40×24)文字，(32×24)文字，(256×192)ドット，(64×48) 画素のうちどれかを使える．

● **画面モード**

画面は常にモード0～3のどれか1つが使われている．分解能や使用コマンドもそれによって異なる．

● **関係演算**

2つの値は等しいか，異なるか，どちらが大きいか小さいかを調べ，答を真（－1）か偽（0）で返す演算．

● **関数**

引数を与えられたら，定められた方法でこれを加工し答として返すもの．

● **キークリック**

Keyclick.キーボードのキーを押すときに出る「プツプツ」という音．

● **キー入力**

コンピュータに対して信号を伝えるための代表的な手段．

● **キーボードバッファ**

キー入力を一時貯え（記憶し）ておくところ．メモリの一部．

● **記憶**

コンピュータの記憶はレジスタとメモリで行なう．電圧の有無を並べて保存し，これを1と0の並びと見て2進数ですべてのデータを蓄積する．

● **機械語**

CPUを直接制御する命令．すべて2進数のコードでできている．

● **キャラクタ**

Character.文字のこと．コンピュータ内では，すべてコードとして扱う．

● **キャラクタコード**

扱うキャラクタに割当てた0～255のコード．

● **行**

BASICの命令を1つ以上並べ，その前に行番号をつけたもの．最大254文字まで許される．プログラム行．

● **境界色**

PAINT文で色を塗るときのわくとなる色．

● **行番号**

プログラム行の前につける番号．0～65529が使え，実行の順序を示す．

● **キロバイト**

Kilobyte．普通1024バイトのこと．

● **組み込み関数**

BASICで用意されているSIN，LEFT$などの関数．

● **グラフィックス**

Graphics.画面に出力される，キャラクタ以外のもの．コードをもたないのでドットの集合として扱う．

● **グラフィックキャラクタ**

日，時，分などの特殊なキャラクタ．ヘッダーとしてCHR$（1）を必要とする．

● **グラフィックマクロ言語**

DRAW文の中で，上下左右などを指定する言語．

● **高級言語**

機械語に比べ人間にもわかり易く，プログラムし易いように作られた言語．FORTRAN，BASIC，PASCALなど．

● **計算**

Z80CPUは，加算，減算，論理演算しかできない．乗算，関数などはこれらを組み合わせた機械語プログラムで行う．

● **固定小数点**

小数点の位置が実際の値の小数点の位置にある数値表現．指数表現を使わない．

● **コマンド**

Command. BASICで与える命令のこと.

● **コマンドレベル**

ダイレクトモードでキー入力（コマンド）待ちになっている状態.

● **コントロールキャラクタ**

Control Character.コードの0～31のキャラクタで,文字ではなく，特殊な働きをするもの.

● **コンパイラ**

Compiler.インタープリタはテキストを解釈しながら実行するのに対し,コンパイラはテキストを一度まとめて機械語に翻訳した後実行する．FORTRANなど.

● **最終参照点**

LP（Last referenced Point)．グラフィックで最後にドットが打たれた，または消された地点.

● **座標**

画面のX，Y座標．左上が（0，0）で右下は画面モードで異なる．通常数学で扱う座標と違ってY軸は下へ向う程値が大きい.

● **座標系**

各画面モードでの座標の設定

● **サブCPU**

メインCPUの補助として専用の機能を下請するCPU．MSXシステムでは音声用にPSG，映像用にVDPが使われている.

● **サブルーチン**

Subroutine.プログラムの中で特定の役割を果す小さな部分．GOSUB文によって繰り返し使われることが多い.

● **算術演算**

加減乗除などの通常の演算.

● **式**

変数や定数，演算子からなる一つの並び．

● **システム**

System. CPU を使うためにメモリ，周辺機器などをつけた全体のものをいう．

● **システム変数**

専用の目的のために用意されている変数．ERR, CSRLIN など．

● **実数**

小数も含むすべての数値．

● **実数型**

小数も扱える変数や定数の型．

● **周辺色**

画面の一番外側の色．表示わくの外で PSET などができない部分の色．

● **周辺機器**

CPU の機能を拡張するための追加部品．プリンタ，ディスクなど．

● **16進数**

0～9とA～Fを数字として数を表現する方法．2進数との変換が楽なのでコンピュータでは多用する．

● **初期設定**

後の作業のために準備すること．MSX システム起動時にはインタープリタにより，BASIC 可動状態が準備される．

● **シリアル**

Serial.一時に1つづつ一連のデータを扱うこと．コンピュータでは1ビットずつデータを入出力するときにいう．カセットテープや RS232C 回線を対象とした入出力がこれに当たる．

● **実行**

プログラムやコマンドがコンピュータによって行われること．

● **真偽**

BASIC では真は 0 以外，偽は 0 という数値で扱われる．

● **数値型**

BASIC の変数の中で数値を代入できるもの．整数型，単精度実数型，倍精度実数型の 3 つがある．

● **スクリーンエディット**

BASIC のエディタによって，画面上で BASIC プログラムテキストを編集すること．

● **スクロール**

Scroll.画面の一番下に新たにテキストを表示するために，それまでに表示されているすべてのテキストが一段上に移動すること．

●**スタック**

Stack.ポインタなどの値を順序よく記憶しておくためのメモリ上の領域.

● **ステートメント**

BASIC 命令の最小単位.

● **ステップ**

Step.プログラム中の命令または行を数えるときに使う言葉.

● **ストリング**

String.文字がつながったもの．文字列．BASIC では255文字までの文字の並びを扱える．

● **ストリングディスクリプタ**

String Discripter.文字列の実体が格納されている番地やその文字列の長さなどを示すポインタ．

● **スプライト**

Sprite.グラフィックパターンのこと．MSX BASIC では 8 × 8，16×16の大きさのものがある．

● **セーブ**

Save.メモリ上のプログラムを外部記憶装置にコピーすること．

● **整数型**

－32768～32767の整数を扱う変数や定数の型.

● **精度**

変数や定数で扱える限界. 単精度型は６桁, 倍精度型は14桁まで正確に表現できる.

● **絶対座標**

X, Yで表される画面上の座標. 左上端を（0, 0）とする.

● **0での除算**

不可能な演算なのでエラーとなる.

● **前景色**

PSET 文, LINE 文などを実行するときに使う色.

● **ソースリスト**

Source List.機械語に翻訳される前のアセンブリ言語（ニーモニックによる), FORTRAN のテキストなどのプログラムリスト. アセンブリ言語では翻訳後の機械語も並記されることが多い.

● **相対座標**

画面上のある点を中心（0, 0）としたときの相対的な座標.

● **添字**

配列変数の何番目の要素かを示すカッコ内の数.

● **ソフトウェア**

Software.一般には利用技術. コンピュータではプログラムのことをいう場合が多い.

● **ターミナル**

Terminal.端末. 中央の大型コンピュータと回線で結ばれたコンピュータ.

● **タイプする**

本書中ではキーを一つ一つ押す動作をこう呼んでいる. 特にキーボードを読むための命令を含むプログラムの実行中以外では, タイプされた文字は画面に表示されるだけなので RETURN キーを押すことを伴う「入力」とは区別している.

● **タイマー**

Timer.コンピュータに内蔵される時計．プログラム中で実行結果の出力などを適度に遅らせるために組みまれたルーチンをさすこともある．

● **ダイレクトモード**

Direct Mode.コマンドを入力と同時に実行したり，行番号をつけたプログラム行を入力したりできるモード．

● **タブレット**

Tablet.→ペンタッチ式のデジタイザ．プログラムによってパネルに描いたものを画面にコピーするときなどに使う．

● **単精度**

有効桁数 6 桁の精度．

● **ダンプ**

Dump.メモリの内容を順に出力したもの．メモリダンプ．

● **中間コード**

BASIC テキストを圧縮して格納するため各命令ごとに決められたコード．

● **データ**

Data.情報．数値，文字，命令など，コンピュータでは最終的にはすべて数値として扱う．

● **データファイル**

プログラム以外のデータの集合．

● **定数**

数字によって表現される一定の数値．

● **ディスク**

Disk.円盤状の磁気記憶媒体．

● **テキスト**

Text.文字によって表現されたもの．文章，それがプログラムを表すとき BASIC テキスト，プログラムテキストなどと呼ばれる．

● **テキストエリア**

BASIC テキストを格納するメモリ上の領域．

● **デジタル**

Digital.不連続量．例えば0と1の間（0.1と0.2の間でも何でもよい）には何もないとする考え方．コンピュータはデジタルな演算しかできないので π や1/3，2の平方根などは近似小数で代用するしかない．

● **デジタイザ**

アナログ量をデジタル量に直す装置．タブレットは手の動き（それによって描かれる線）というアナログ量をX，Y座標というデジタル量に変換する装置である．

● **デバイス**

Device.→周辺機器．

● **デバッグ**

Debug.プログラム中の間違い（バグ，bug）を取り除き正しいものにする作業．

● **デフォルト値**

Default.システムによってあらかじめ定められた値．コマンドのパラメータを省略したときなどにこれが適用される．

● **特殊記号**

！，％，＄など BASIC で特別な意味をもつ記号．

● **ドット**

Dot.グラフィックスを扱う上での最小の単位．画面上の1点．

● **内部表現**

→中間コード．

● **流れ図**

プログラムの流れを描いた図．フローチャート．

● **ニーモニック**

Mnemonic.アセンブリ言語でプログラムを書くときに使う単語．機械語と一対一の対応となる．

● **2進数**

1と0だけで数字を表現する方法．

● **2の補数**

2進数で負の数を表すのに使われる．

● **入力する**

コンピュータに命令やデータの信号を送ること．キーボードのキーを押すことは機械語レベルでは入力といえるが，BASICのコマンドを実行したり，INPUT文に応答したりするためには RETURN キー入力を必要とするので，これらは本書では単にキーを押す「タイプする」と区別して「入力する」と呼んでいる．

● **ヌルストリング**

Null String.長さが0の文字列．〝〟またはCHR$（0）

● **ノイズ**

Noise.音階以外の音．

● **ハードウェア**

Hardware.機械，機材そのもの．コンピュータシステムではプログラム以外のすべて．

● **背景色**

PRESET文で使われる色．画面の地の色．

● **倍精度**

有効桁数14桁の精度．

● **バイト**

Byte. 8ビット． 0～255を表せる情報の単位．

● **バイナリ形式**

Binary.コンピュータメモリの内容となっている数値をそのまま入出力する形式.

● **配列**

変数の束，添字を使って必要な要素を指定できる.

● **バグ**

Bug.プログラム中の誤り.

● **8進数**

0～7の数字だけを使った数値表現.

● **バッファ**

Buffer.コンピュータがデータの入出力を行うときに，一度データをいくつかまとめて置いておくところ．通常メモリの一部を使う.

● **パラメータ**

Parameter．命令に付属する数値，文字などの値.

● **パラレル**

Parallel.一時に複数のデータを並列に扱うこと．コンピュータではたとえば8ビットずつデータを入出力するときにいう．セントロニクス仕様のプリンタ出力など.

● **番地**

メモリの一単位に当てられた番号．Z80では0～65535（16進で&H0～&HFFFF)

● **比較演算**

2つの値の大小，等価を調べ真（－1），偽（0）を返す演算.

● **引数**

関数に与えられる値．文字型と数値型がある．関数はこれを加工して結果を返す.

● **ビット**

Bit.情報の最小単位．1か0，＋か－など．コンピュータはこれを電気信号で扱いすべての動作の基本とする．Z80は8ビットを同時に扱うことができる.

● **ビット操作**

8ビット（または16ビット）の並びの中の任意のビットをON／OFFするような操作.

● **ブール演算**

論理代数学の演算.「本当の本当は本当，本当の嘘は嘘」などを数の演算として扱える.

● **ファームウェア**

Firmware.ソフトウェアの中でも特にシステムの基礎となるもの. ROMに書き込まれている場合が多く,MSXシステムではBASICインタープリタプログラムのこと.

● **ファイル**

File.情報（データ）の集合. コンピュータではプログラムファイルとデータファイルに大別する.

● **ファイルコントロールブロック**

BASICがファイル入出力を管理するための一種のワークエリア. バッファの位置などの情報が書き込まれる.

● **ファイルディスクリプタ**

BASICが扱うファイルがどのデバイスにあり，何というファイル名をもつかなどを示す文字列.

● **ファイル番号**

BASICでファイルを扱うときに割り当てる番号. OPEN文で割り当て，PRINT #文などで入出力を行う.

● **ファイル名**

BASICで扱うファイルにつけた名前.

● **フォーマット**

Format.ディスクをシステムに適した状態に初期設定すること.

● **フォント**

Font.表示文字の構成，ドットの並び方.

● **符号ビット**

BASICの整数は16ビット（2バイト）以内で表現される．このときマイナスの符号としてMSBが使われる．

● **浮動小数点**

実数の中で指数表現を用いたもの．この小数点は絶対的ではない．例えば3.14E＋4は31.4E＋3と同じである．

● **プレーン**

Plane.→仮想画面．

● **フローチャート**

Flowchart.→流れ図

● **プログラム**

Program.コンピュータに対する命令の集り．機械語，BASIC, FORTRANなどの言語で書かれる．

● **プログラム行**

BASICプログラムの一行，最大254文字まで．

● **プログラムファイル**

プログラムも情報（データ）だがコンピュータでは特に重要なのでそれ以外のデ--タファイルと区別してこう呼ぶ．

● **プログラムモード**

BASICでRUN, GOTOなどのコマンドにより，記憶されているプログラムが実行されているモード．

● **プロンプト文**

INPUT文などに使う．入力を促すようなメッセージのこと．

● **文**

→ステートメント．

● **分解能**

グラフィックスで1画面にどれだけ多くの点が使えるかということ．MSX BASICのSCREEN2なら256×192となる．解像度ともいう．

● **分岐**

プログラムの中で条件によって制御の流れが変わる部分．IF〜THEN〜など．

● **ヘッダー**

Header．ファイル，テープ，何かの信号などの先頭となる部分．グラフィックキャラクタにもCHR$(1)というヘッダーが使われている．

● **変数**

必要に応じて値を代入したり，式の一部になったり，関数の引数になったりできるもの．代数．

● **変数名**

変数として使う名前．MSX BASICでは英字1字または英字1字＋英数字1字．予約語を含んではならない．

● **変数領域**

BASICのメモリの中で，変数名やその型，値が格納される部分．

● **ボーレート**

Baud Rate.信号を送受信する速さ．1秒間に100ビット送るなら100 bpsと呼ぶ．

● **ポインタ**

Pointer.メモリ上の何か意味のある内容をもった番地を示すもの．

● **暴走**

プログラムの誤りによって無限ループなどになり，コンピュータが予定外の動作をする状態．BASICのプログラムなら常に CTRL ＋ STOP キーが有効だが機械語では電源を切るなどしなければ止められないことが多い．

● **マルチステートメント**

ステートメントをコロンでいくつかつなげたもの．

● **ミュージックマクロ言語**

PLAY 文のパラメータとなる文字列で，音程，音長などを使って音楽をプログラムするための言語.

● **無限ループ**

永久に続くループ.

● **メインルーチン**

Main Routine.プログラムの大きな流れを形づくる部分.

● **メガバイト**

Megabyte.1024キロバイト．2^{20} バイト.

● **メモリ**

Memory.コンピュータの記憶装置．IC 群からなる.

● **文字型**

変数や定数で数値でなく，文字列を扱う型.

● **文字列**

→ストリング

● **文字列演算**

文字列を連結したり，分割したりする演算.

● **ユーザー**

User.コンピュータを使う人.

● **ユーザー定義関数**

ユーザーが定義する関数. DEFFN 文を使う．ルーチンを機械語で組めば USR 関数も使える.

● **有効桁数**

正確であると保証できる数値の桁数．単精度なら 6 桁，倍精度なら14桁まで.

● **予約語**

BASICのコマンド，関数，ステートメント演算子に使われる単語すべて．キーワード．

● **乱数**

でたらめな数，数の並び．BASICでは凝似乱数を計算で作り出して使っている．
RND(1)の形で0～1未満の乱数を得る．

● **ルーチン**

Routine.プログラムのまとまった一部分．または小さなプログラムの全体をいうこともある．

● **ループ**

Loop.プログラム中の繰り返し処理．

● **レジスタ**

Register.CPU内部の記憶装置．

● **ロード**

Load.外部記憶装置上に保存されているプログラムをメモリにコピーすること．

● **論理演算**

ビット操作，ブール演算などに使われる演算．MSX BASICでは16ビット単位に行われる．

● **ワークエリア**

Workarea. BASICインタープリタがメモリ上に使用する作業領域で，テキスト，変数などの管理を行ったり，バッファとして使ったりしている．

● **ワード**

Word. 1つの単語．16ビット（2バイト）を1ワードと呼ぶこともある．

● **割り込み**

プログラムの通常の流れを，外的要因によって強制的に変えること．